Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Solicito: aplicativo Android para apoio à logística de transportes terrestres

Victor Canato

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Orientador
Prof. Dr. Wilson Henrique Veneziano

Brasília
2019

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Curso de Engenharia da Computação

Coordenador: Prof. Dr. José Edil Guimarães de Medeiros

Banca examinadora composta por:

Prof. Dr. Wilson Henrique Veneziano (Orientador) — CIC/UnB
Prof. Dr. Jorge Henrique Cabral Fernandes — UnB
Prof.ª Dr.ª Germana Menezes da Nobrega — UnB

CIP — Catalogação Internacional na Publicação

Canato, Victor.

Solicito: aplicativo Android para apoio à logística de transportes terrestres / Victor Canato. Brasília : UnB, 2019.
110 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2019.

1. Android, 2. Java, 3. Logística, 4. Transporte Terrestre

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Solicito: aplicativo Android para apoio à logística de transportes terrestres

Victor Canato

Monografia apresentada como requisito parcial
para conclusão do Curso de Engenharia da Computação

Prof. Dr. Wilson Henrique Veneziano (Orientador)
CIC/UnB

Prof. Dr. Jorge Henrique Cabral Fernandes
UnB

Prof.ª Dr.ª Germana Menezes da Nobrega
UnB

Prof. Dr. José Edil Guimarães de Medeiros
Coordenador do Curso de Engenharia da Computação

Brasília, 30 de Janeiro de 2019

Dedicatória

Minha dedicatória vai para todos que veem depois de mim, tenham seus objetivos, mas aproveitem a vida. Para quem está lendo meu trabalho deixo alguns aprendizados que tive durante minha longa jornada na Universidade, até por que a Universidade pode ser seus piores 5 anos ou melhores 10, no meu caso 9. Nossas vidas giram em torno de problemas. E sabendo lidar com esses problemas sua vida pode ser muito melhor, qual seria o objetivo da vida se não ser feliz? Tive 3 frases que me acompanham desde sempre que gostaria de dedicar a todos que cruzarem meu caminho, a primeira fez uma rápida reflexão sobre o que é ser feliz "*Uma vida feliz não é uma vida sem problemas... Uma vida feliz é uma vida com problemas bem resolvidos*", parando e refletindo, se tivéssemos tudo de mão beijada, qual seriam nossos objetivos? Mas não adianta apenas sabermos que temos que resolver os problemas, ai que entra a segunda frase "*Desculpas não resolvem problemas. Soluções e projetos sim*", para se resolver um problema precisamos pensar sobre ele, sejam sempre criteriosos, curiosos, procurem aprender tudo que possam com cada experiência e não aceitem tudo que impuserem, procurem entender. Somente após a compreensão do seu problema a terceira frase se encaixará "*Existem dois tipos de problemas: Se eles tem solução, não precisa se preocupar. Se eles não tem solução, não tem por que se preocupar*", desta forma poderá ter uma vida leve, bem resolvida e o mais importante, feliz. Ainda deixo aquele ultimo conselho, que eu não segui por 24 anos da minha vida, tenham um objetivo e mantenham o foco nele, esse é o segredo do sucesso.

Agradecimentos

Gostaria de agradecer a todos que me ajudaram a chegar onde cheguei e todas as experiências que me proporcionaram, meu Centro Acadêmico de Engenharia da Computação no qual passei 3 ótimos anos na gestão(contando todos meus poucos veteranos e diversos calouros), meu primeiro grupo de Diretório Central dos Estudantes que tanto me ensinaram sobre movimento estudantil, ao meu time de futebol americano BrasíliaV8 onde encontrei uma família de muita zueira e ótimos jogos, Dança, pessoal da Biologia, UnBaladas, Primato, Maquinada(org e bateria), Comissão eleitoral do DCE, meus pais e família, Vestibular cidadão, Google, Wikipédia, wolfram alpha, copular, ACinfo, aliança etc..

CAEC Pacheco, Bispo, Angelita, Igão, Matheuzinho, Coletti, Isaac, Pigatto, Fraps, Uriarte, Yan, Barreto, Bruxão, Malzoni, panda.

DCE Raul, Sebba, Moretti, Cuia.

Bio Babi, Redorat, Texugo, Stifler, Herik, Paraca, Luci, Yakult, Marcelinho, Sec, Fê, Don, Larica, Radan, Pri, Deus, Bar.

V8 Bruninho, Raphão, Magrão, Mario, Luizinho, Pedrinho, Txe, Bau, Kiush, Nachos, Budo, Shaq, Px, Caverna, Toad.

UnBaladas Macarrão e Primo.

Maquinada Andrezinho, Thamires, Yu, Kildery, Cissa, Mattos, Bia, Cissa, Juca, Jessica, Karol, David, Iguinho, Pudim, Andrey, Victor Hugo, Franco, Ciro, Gago, Bragança.

VC Papalardo, tigresa, Ian, Gabriel, Ana.

Copular Kpta, Lucas, Leo, Marcelo, Cae, Imperatriz, João e Limão.

ACInfo Alê, Nati, Sena, Leo, Malta, Jessica, Tainah, Pirineus, Pri, Emily, Josian, Weber, Renan.

Aliança Pedro Ivo, Nicholas, Gabi, Kkxo, Zaponni, Bertonni, Sophia, Sorriso.

Shazam Fabricio, Ian, Marcelo, Walber, Pepinho.

Família Júlio, Tila, Vani, Renata, Raul, André, Sílvia.

Perdidos no rolê Lukão, Bia Vieira, Inseto, Kaya, Paula, Dani, bill,

Agradecer a AIESEC que me deu a oportunidade de morar na europa e trabalhar com desenvolvimento para celular, o que me fez me reencontrar com meu gosto por programação

Preciso fazer um agradecimento especial, nesse projeto tive apoio do meu orientador Wilson. Que mesmo eu sendo um aluno que deu uma certa dor de cabeça na matéria de desenvolvimento de aplicativos se propôs a me orientar a distância e me proporcionou a chance de fazer uma tese desenvolvendo o que me da prazer. E agradecer pela matéria por ele ministrada que me abriu portas e mudou minha vida.

E a minha namorada, pode parecer simples, mas se não fosse por ela, se não fosse por meu objetivo de ter um futuro com ela talvez eu não tivesse foco e vontade para sentar, pesquisar, escrever e produzir. Mais de uma vez pensei em largar e seguir em outra direção, mas o pensamento nela que me manteve firme. Me motivou a ir para fora e buscar sonhos maiores e fico feliz de fazer isso ao seu lado. Agradeço por me fazer companhia e incentivar, mesmo sem perceber, até ela dormindo, enquanto eu virava a noite produzindo essa monografia, era uma forma de me incentivar. A presença dela além de sempre me alegrar me deu um norte. Obrigado Vida, Mayumi Pachecho Hamaoka.

Estou aqui para provar que frito também forma!

Resumo

Este trabalho propõe resolver um problema de plano cartesiano onde se possui diversos endereços iniciais e finais e se tem intenção de analisar as distâncias e tempo de trajeto entre a lista de endereços iniciais e finais.

Para isso foi desenvolvido um aplicativo para telefone celular que auxilia na logística de transportes terrestres. A intenção final do programa é evitar o trabalho de análise par a par automatizando o processo.

Foram levantados requisitos junto a profissionais que atuam nesse ramo de mercado. O aplicativo proporciona ferramentas de automatização, podendo salvar listas de locais de partida e de pontos finais, mostrando todas as combinações entre os pontos com medidas de tempo e distância. Ainda, pode se utilizar a localização atual do usuário para facilitar mudanças repentinas no trajeto, tudo isso com a praticidade do sistema no celular.

O produto final possui características de um produto viável mínimo. O aplicativo foi testado por profissionais que atuam no mercado de trabalho na área de logística terrestre. Os resultados finais mostraram que a ferramenta conseguiu automatizar o calculo de distância e tempo para diversos endereços.

Palavras-chave: Android, Java, Logística, Transporte Terrestre

Abstract

This paper proposes to solve a problem of Cartesian Plan where we have several initial and final addresses and intends to analyze the distances and time of route between the list of initial and final addresses.

For this, a mobile phone application was developed that assists in the logistics of land transportation. The final intent of the program is to avoid the parsing work by automating the process.

Requirements were raised with professionals working in this market segment. The application provides automation tools, which can save lists of starting and ending points, showing all combinations of points with measures of time and distance. Also, the user's current location can be used to facilitate sudden changes in the route, all with the practicality of the system in the cell phone.

The final product has characteristics of a minimum viable product. The application has been tested by professionals working in the labor market in the area of land logistics. The final results showed that the tool was able to automate the calculation of distance and time for several addresses.

Keywords: Android, Java, Logistics, Land Transportartion

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A ideia do desenvolvimento do *software* surgiu durante um ajuda prestada para outra monografia [1] qual fez um estudo sobre os melhores portos para exportação internacional. A conclusão foi que as soluções utilizadas hoje não são ótimas. Durante esse estudo e pesquisa foi necessário correlacionar diversos portos brasileiros com o local de armazenamento ou produção dos produtos. O que se mostrou uma tarefa repetitiva e custosa em questão de tempo.

Buscou-se um aplicativo de celular ou softwares que pudessem automatizar a análise dos endereços dos portos e dos armazéns, as buscas foram feitas em lojas onlines como *Google Play Store* e *Apple Store*, em seguida foi buscado na base de dados Periódicos da CAPES, nada foi encontrado. Utilizou-se as palavras-chaves: Logística Terrestre, caminho, rotas, lista de endereços, transporte, melhores rotas com diversos pontos. Tanto em português como em inglês. Nesse momento percebeu-se uma lacuna de programa computacional com foco em auxiliar a correlação de distância e tempo entre diversos pontos.

1.1 Problema

Apesar de termos fácil acesso ao *Google Maps* tanto em computadores (*Desktops, Laptops*) como em dispositivos móveis (celulares e *tablets*) esse sistema operacional apenas gera a correlação entre um ponto inicial, um final e possíveis pontos intermediários. Como podemos observar Figura 1.1.

Porém caso queiramos ter um comparação entre Figura 1.1, Figura 1.2 e Figura 1.3 seria necessário fazer diversas combinações para um ponto inicial, se tratando de um trabalho de N*1, agora se ainda tivermos a opções de vários pontos iniciais teríamos então um trabalho de N*N.

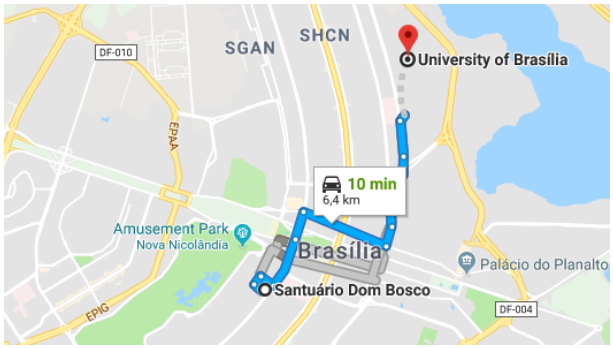


Figura 1.1: Trajeto via Google Maps do Santuário Dom Bosco à Universidade de Brasília (Fonte: [2]).

Figura 1.2: Trajeto via Google Maps do Santuário Dom Bosco à IESB (Fonte: [2]).

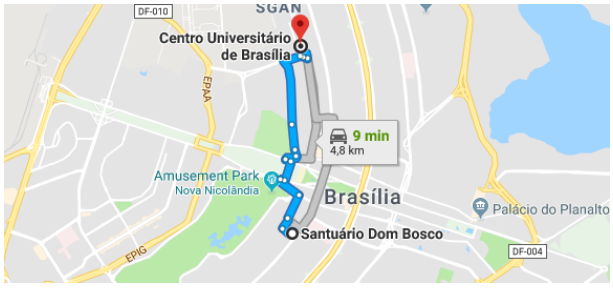


Figura 1.3: Trajeto via Google Maps do Santuário Dom Bosco à CEUB (Fonte: [2]).

1.2 Justificativa

Hoje existe uma necessidade de agilidade para a solução de problemas, os dispositivos móveis hoje encontrados no mercado contam com diversos aplicativos que respondem rapidamente e estão na palma da mão possibilitando um trabalho com maior eficácia e rapidez ao solucionar problemas mesmo enquanto nos locomovemos ou estamos fora do escritório.

1.3 Objetivo Geral

Construir um software para dispositivos móveis que ajude a solucionar problemas de logística enfrentadas por empresas na hora do planejamento e execução. Um aplicativo que salve listas com diversos endereços e consiga analisar todas as rotas entre duas listas mostrando tempo e distância. Podendo exportar o resultado para navegação.

1.4 Objetivos Específicos

Para alcançar esse objetivo geral, foram estabelecidos os seguintes objetivos específicos para as funcionalidades do aplicativo:

1. Criar listas extensas de endereços/locais;
2. Os endereços salvos e utilizados precisos;
3. Observar a relação de tempo e distância entre os endereços das listas;
4. Apresentar o trajeto de dois pontos para navegação terrestre;
5. Utilizar localização atual para comparação com lista de endereços;
6. Salvar listas na nuvem em banco de dados para portabilidade entre aparelhos;
7. Ter uma opção de uso sem necessidade de autenticação.

1.5 Resultado Esperado

Espera-se que o aplicativo ajude os usuários da área de logística onde, por exemplo, eles possuam vários endereços de armazenamento e diversos portos para exportação, facilitando a escolha do melhor porto em relação a cada armazém. E que também possa auxiliar agentes em campo da empresa que podem precisar se deslocar para um ponto de apoio da empresa e necessita decidir entre diversos endereços de uma lista extensa qual deles é o mais próximo.

Sabendo ainda que as possibilidades nessa área de logística são de um alto alcance é de interesse que o código seja o inicio de um projeto *Open-Source* que continuará a se desenvolver com as análises e sugestões dos usuários.

1.6 Metodologia

O processo se deu primeiro em ajudar a solucionar o problema da monografia sobre o Porto do Itaqui sobre todas as possíveis combinações de distância e tempo entre os endereços, assim foi possível entender quais são as soluções existentes e onde existe uma carência de aplicativo para essa automação. Após isso foram feitas entrevistas online com trabalhadores na área de logística de transporte em uma empresa multinacional para entender quais são as principais características e funcionalidades do aplicativo desejadas.

Depois uma pesquisa foi feita procurando por aplicativos que tenham soluções parecidas e o que já é disponibilizado evitando que ouve-se repetição de trabalho já desenvolvido. Após a conclusão do desenvolvimento foram feitos testes.

Para aplicação dos testes e na falta de um grupo profissional de testadores foi selecionado um grupo de usuários betas com pessoas da área de logística terrestre para validação do produto final. Durante o período de testes, para garantir um uso contínuo, foram enviadas notificações diárias indicando funcionalidades para se testar e ao final do período de testes um questionário foi respondido pelos usuários.

1.6.1 Plataforma

Será explicado na sessão 2.2 sobre o crescimento da utilização dos aplicativos em *smartphones* e baseado no *site* StatCounter o sistema operacional *Android* é o mais utilizado em equiparação com qualquer outro sistema, como podemos observar na Figura 1.4.

A *Android, Inc.* foi fundada em 2003, na Califórnia. Posteriormente, em 2005, foi adquirida pela *Google TM* e, em 2007, foi lançada a primeira versão do sistema. O primeiro *smartphone* disponível comercialmente rodando o sistema foi o HTC Dream. Desde então, o Android vem recebendo diversas atualizações e desenvolvimento de novas funcionalidades.

Sendo assim o *Android* foi o sistema operacional escolhido para o trabalho, mesmo que fosse ideal um projeto de multiplataforma é inviável nesse momento tentar abranger todas as opções possíveis.

1.6.2 Ferramentas de Desenvolvimento

Para o desenvolvimento do aplicativo *Android* foi decidido utilizar a *IDE Android Studio* e a linguagem *Java*. Mais informações serão explicadas na seção 3.2.2.

Outra ferramenta muito utilizada no projeto foi o *Firebase*, esse foi incrementado ao projeto para autenticação, segurança, banco de dados e relatório de erros. Para garantir mais precisão nos endereços e locais foi utilizado a API do Google Maps. E para distri-

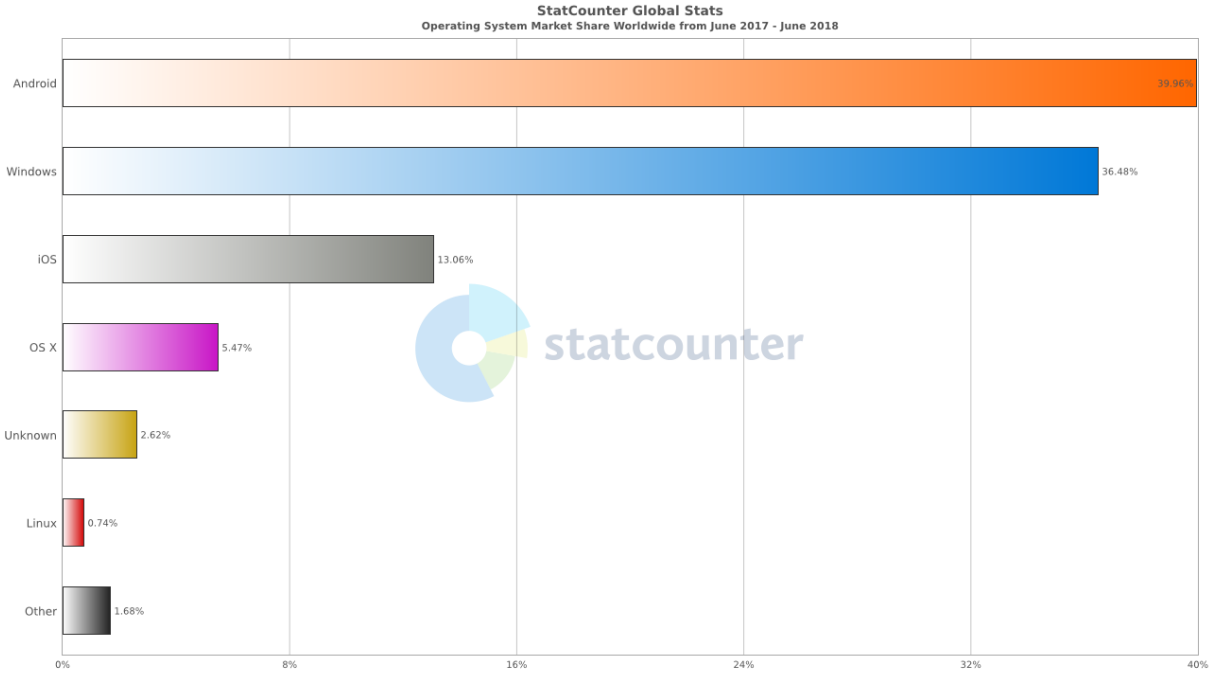


Figura 1.4: Uso Mundial de Sistema Operacional (Fonte: [3]).

buição o aplicativo foi disponibilizado na Google Store (Google Play) facilitando o uso, download e trazendo mais confiabilidade ao cliente final.

Capítulo 2

Referencial Teórico

As referências foram separadas em duas seções uma focada no motivo da pesquisa, que usou como ponto de partida o trabalho de graduação de outra aluna da UnB como citada na introdução.

E a segunda seção fala sobre o aplicativo em si e por que a escolha da plataforma para desenvolvimento.

2.1 Logística Terrestre

A logística terrestre passa por diversos pontos importante como custo e tempo, segundo Ballou [4] o transporte de mercadorias é um componente do sistema logístico e possui algumas atividades principais em seu escopo. A seleção do modal e determinação de roteiros são uma delas e se inserem no tópico da pesquisa.

O transporte de mercadorias representa uma das maiores despesas dentro da logística empresarial [4]. Os custos na área de transporte são definidos pelas despesas com a movimentação entre dois pontos, manutenção e gerenciamento do estoque em trânsito [5].

2.1.1 Portos, Armazéns e Fábricas no Brasil

Levando em consideração somente a amostra do assunto da tese que incentivou essa pesquisa temos ainda um cenário gigante.

Hoje no Brasil temos 37 Portos Públicos organizados no país, 122 Portos Fluviais, 235 instalações portuárias [6] totalizando 394 destinações finais.

Em 2013 foram contabilizadas 312 mil industrias no Brasil, grande maioria de alimentação [7]

E o Companhia Nacional de Abastecimento possui registrado 17,7 mil armazéns no país [8] [9]

Nesse cenário restrito e ainda extremo em números teríamos mais de 2 trilhões de opções.

Pela pesquisa Análise do potencial de demanda para exportação pelo Porto do Itaqui sabemos que as soluções ótimas não são usadas por que não são encontradas, por isso o foco desse aplicativo de facilitar o encontro das melhores rotas terrestres dado muitos pontos.

2.2 Aplicativos para Celular

Em 1983 [10] Steve Jobs já previa o que viriam a ser os aplicativos e essa é uma das maiores revoluções que estamos vivendo. Mas os aplicativos como conhecemos foram lançados em Julho de 2008 com a App Store com 552 aplicativos. Com uma semana foram 10 milhões de Downloads, com 60 dias 100 milhões e 3 mil aplicativos para download em 62 países diferentes.

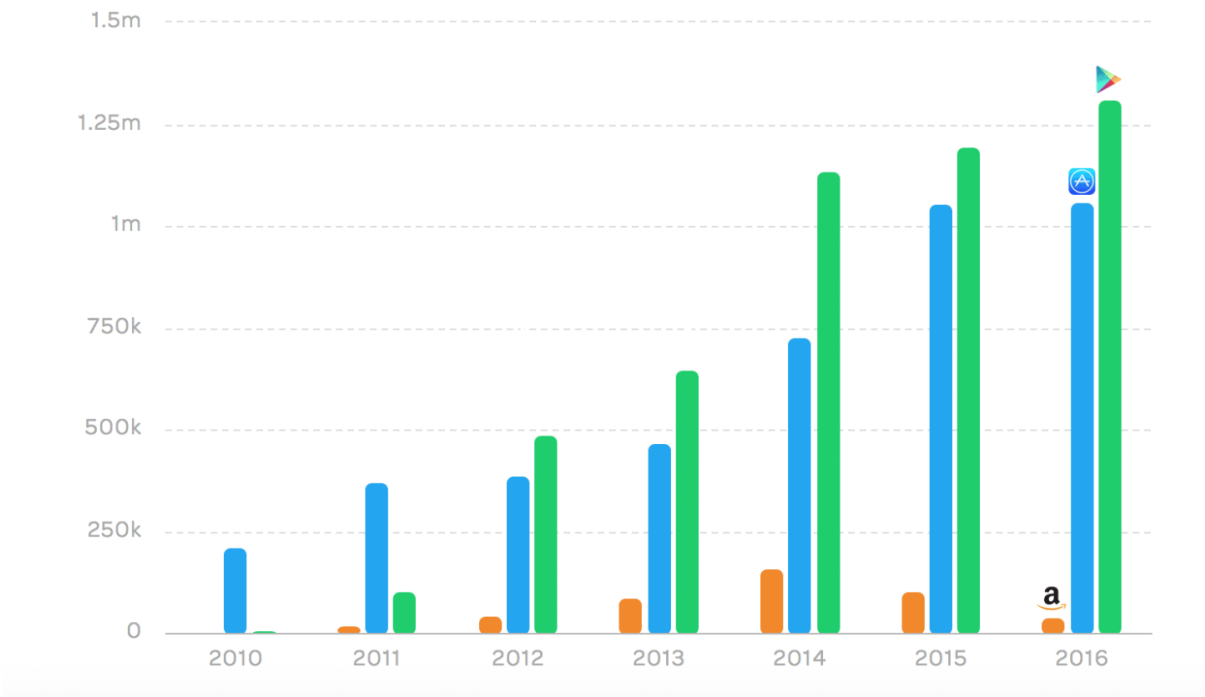


Figura 2.1: Número de novos aplicativos lançados por ano (Fonte: [11]).

Em Outubro de 2008 a Google lança a loja de aplicativos para Android. No final de 2009 Angry Birds e Whatsapp foram lançados. Entre outros lançamentos como Window Store e Amazon Store a palavra *app* foi votada como a palavra do ano em 2010 e foi calculada a geração de quase 300 mil empregos nos Estados Unidos ligados diretamente ao desenvolvimento para iOS até o final de 2011. [12]

Tudo isso afirma o tamanho e impacto que os aplicativos tem na nossa sociedade e como devem ser estudados e focados não apenas em cursos de tecnologia, mas em qualquer área de estudo e vivência. As implicações dos aplicativos hoje ainda não são totalmente conhecidas e sua evolução é uma das mais rápidas da história. Como no caso do aplicativo *Flappy Bird* que foi retirado pelo seu criador por medo de viciar os usuários em larga escala. "13 kids at my school broke their phones because of your game, and they still play it cause it's addicting like crack." [13] Mas também temos bons exemplos como quando a BBC, empresa de rádio de Londres, usou o whatsapp para espalhar informações e combater a ebola em Outubro de 2014 [14]

E os números continuam crescendo de forma assustadora, usuários, dinheiro, downloads, aplicativos. Como podemos observar na Figura 2.1 o número de novos aplicativos cresce todo ano.

2.2.1 Software Portátil

A decisão de desenvolver um software que pudesse estar a fácil alcance se deu por uma característica desejada pelos usuários que foi levantada durante as entrevistas nas empresas. O aplicativo deve poder se adequar ao dia a dia, caso por exemplo um dos endereços de visita seja cancelado o usuário pode rapidamente usar sua localização para descobrir o mais próximo.

Ou ainda digamos que o agente precise se locomover para um dos postos avançados ou lojas da rede, basta ele selecionar a lista de lojas desejadas e ver qual delas está ao seu alcance de forma mais eficaz.

2.2.2 Crescente Uso de *Smartphones*

Não é nenhum segredo que hoje os usuários comuns já não compram mais computadores como 10 anos atrás. O que antes era esperado se encontrar uma CPU dentro de casa para a família e depois se transformou em laptop para alguns integrantes, hoje o celular supre as necessidades do dia a dia da maioria da população, passando a agir como computador pessoal.

Dados do IBGE [15] apontam o crescimento de celulares no Brasil, onde 3/4 das pessoas com mais de 10 anos possuem celular, sendo que do restante que não possui metade está entre 10 e 17 ou tem mais de 60 anos. Sem contarmos ainda pessoas com mais de um celular ou com mais de uma linha no mesmo aparelho. Em contra partida os chamados microcomputadores (*dektops e laptops*) estão presentes para menos da metade da população. E ainda podemos indicar a presença de tabletes em 1/6 dos Brasileiros.

Capítulo 3

Desenvolvimento

O desenvolvimento foi separado em 3 principais categorias: o processo que descreverá como o projeto foi pensado e priorizado para incrementar todas as funcionalidades; a arquitetura qual ditará a organização dos arquivos, códigos e como eles se comunicam; e por ultimo as tecnologias usadas desde linguagens até distribuição. Ainda neste capítulo contamos com os requisitos que tratam sobre o público alvo e as funcionalidades do aplicativo.

3.1 Processo

Como definido [16] o processo de desenvolvimento de software é a ação de separar as etapas de construção em fases distintas que visam incrementar o *design*, o gerenciamento do produto e do projeto. Pode ser conhecido também como o ciclo de vida do desenvolvimento do software.

Existem algumas formas diferentes de realizar o processo de desenvolvimento como cascata, espiral e o que foi usado nesta monografia que é o desenvolvimento ágil. Como Scott Ambler define [17] é mais importante o trabalho em si do que como ele é feito ou documentado, porém não é exatamente um afronta a metodologia ou a documentação, mas uma balança, parafraseando e traduzindo do inglês Jim Highsmith sobre a história no site do Manifesto Ágil [18]:

3.1.1 Metodologia Ágil

"O movimento Ágil não é anti-metodologia, na verdade muitos de nós querem restaurar a credibilidade da palavra metodologia. Queremos restaurar um equilíbrio. Nós adotamos a modelagem, mas não para arquivar alguns diagramas em um repositório corporativo empoeirado. Nós adotamos a documentação, mas não centenas de páginas de volumes

mantidos e raramente usados. Nós planejamos, mas reconhecemos os limites do planejamento em um ambiente turbulento. Aqueles que classificariam os proponentes de XP, SCRUM ou qualquer uma das outras metodologias ágeis como "hackers"ignoram tanto as metodologias quanto a definição original do termo hacker."

Tendo esses valores bem definidos foi extraída a parte que mais interfere diretamente nesse aplicativo, sendo que normalmente essas metodologias tratam de desenvolvimento para grupos de pessoas e não uma programação singular. Sendo assim foram utilizado os elementos de organização da metodologia. Como preparação da *sprint* (próximas duas semanas de trabalho) com objetivos e atividades bem definidas, durante o processo caso surgisse novas propostas de funções ou correções de código eram inseridas no *Backlog* do projeto com descrição, ao final da *sprint* se fazia uma análise do trabalho efetuado e se atingiu a meta, para depois análisar o *backlog* e organizar as prioridades das próximas semanas de trabalho.

O código então deverá a cada incremento ter seu produto final funcional. Evita-se assim pedaços de código inacabado ou sem uso.

Por exemplo, na versão inicial do aplicativo não era possível autenticar, por isso apenas duas listas de endereços eram disponibilizadas, com nomes estáticos e já criadas populadas ou não. Depois de criar a autenticação se fez necessário deixar o usuário criar quantas listas lhe fossem necessárias. Para isso foi preciso desfazer as duas primeiras listas, colocar opção de criar listas e verificar se o usuário está logado ou não para saber se salva localmente ou no banco de dados. Esse tipo de trabalho apesar de parecer dobrado na verdade garantiu versões funcionais do aplicativo que se espalhavam por meio do uso de *git* em diversos galhos do desenvolvimento. Além de ajudar também para o processo de resolver um problema por vez onde é possível fazer primeiro a localização funcionar e depois como salvar isso no banco de dados e não ficar pensando em como vai ser a melhor solução para todos os casos e depois ver o funcionamento final.

3.2 Tecnologia

É interessante observar que as tecnologias usadas no trabalho fazem todas parte da mesma empresa Google Inc. o que gera a reflexão interessante sobre o monopólio de empresas. Google hoje oferece diversas ferramentas prontas que facilitam a vida do usuário e em troca consomem todos os dados que passam por essas ferramentas e caso o uso seja grande cobram posteriormente, apesar de ter uma opinião favorável a esse tipo de ação é assustador pensar que somente a Google é mantenedora de quase metade dos servidores mundiais.

3.2.1 Android Studio

Apesar de existirem outras opções, hoje o *Android Studio* com base em *JetBrains' IntelliJ IDEA* é certamente a melhor ferramenta para desenvolvimento de aplicativos android. Já possui integração com Firebase e dispões dos mais novos métodos da linguagem. Com editor de layout, emulador de celular, análise de código e suporte a Java e Kotlin.

Além de ser a IDE (*Integrated Development Environment*) oficial do *Android*, construída especificamente para o desenvolvimento de aplicativos, é gratuita para uso.

3.2.2 Linguagem

Os aplicativos desenvolvidos para *Android* podem ser desenvolvidos de forma direta em três linguagens e ainda podem conter as três linguagens juntas no mesmo projeto: Java, Kotlin e C++. O desenvolvimento é feito no *Android Studio*.

O aplicativo foi desenvolvido na linguagem Java e para interface e *layout* foi utilizado XML que é o padrão.

Apesar do Kotlin ser a nova promessa da Google para desenvolvimento Android a linguagem ainda não tem o número de guias, perguntas, respostas e usuários que a linguagem Java. Também é sabido que uma troca de Java para Kotlin é normalmente fácil e não apresentaria grande trabalho no futuro caso fosse necessário.

3.2.3 Firebase

Projeto desenvolvido pela Firebase Inc. e adquirido pela Google em 2014 traz diversas soluções para Android com três pilares principais: Construir aplicativos melhores, melhorar a qualidade dos aplicativos e fazer o uso do seu aplicativo crescer.

Para cada uma das tarefas o Firebase tem mais de 15 soluções e hoje é usado por diversos aplicativos mundiais. Vamos focar então nas funcionalidades importantes para este trabalho.

- Authentication, para acesso dos usuários;
- Cloud Firestore, para salvar listas e dados;
- Crashlytics, com relatóriosc de mal funcionamento, congelamento do app e quebras;
- Google Analytics, para relatório de usos, tempo, onde, público, crescimento, etc;
- Cloud Messaging, que envia notificações para usuário no celular, usado para lembrar os usuários dos testes durante o período de validação.

3.2.4 Google Maps Platform

Essa plataforma foi usada para garantir uma melhor veracidade dos endereços no aplicativo. O Google Maps hoje contem informação ativa de diversos usuários por GPS, geolocalização, fotos, traçando rotas, confirmando locais, nomes e ainda utilizando o Google My Business garantindo o local da empresa por meio de validação.

Tendo 3 principais divisões hoje o Google Maps conta com Maps que mostra o mapa em si com visão de desenho ou fotos de satélite, Places que ajuda a descobrir lugares, achar lojas e verificar qualidades e, o nosso foco, Routes que entrega localização de endereços e traça rotas para locomoção.

Routes possui opções como caminhar, bicicleta, transporte público e direção, no nosso estudo só nos importa a ultima opção que é a que o usuário guia um automóvel, porém podemos ver fácil adaptação para as outras opções o que aumenta o leque do aplicativo.

No caso do aplicativo como estamos considerando não só a distância, mas também o tempo de deslocamento o uso de uma ferramenta como o Google Maps se torna indispensável pois ele possui informações sobre trânsito, congestionamento, acidentes e consegue calcular outras rotas melhores ou atualizar o tempo de demora para completar o percusso.

3.2.5 Play Store

Por último temos a loja virtual Google Play Store que é a mais conhecida e usada para distribuição, venda, busca e avaliação de aplicativos Androids. Sua facilidade se da principalmente pelo costume do usuário Android com essa loja virtual visto que todos os celulares com esse sistema operacional já possuem essa loja instalada no celular por padrão.

Além disso hoje a loja conta com 4 diferentes métodos de distribuição:

- Internal test track, utilizado para testar com pessoas próximas, normalmente dentro da própria empresa, necessita cadastrar um grupo de e-mails. Lança o app na mesma hora que envia o arquivo;
- Close track, utilizado para testar com um grupo de testes fechado com vários usuários selecionados. Necessita cadastrar um grupo de e-mails. Lança o app de forma rápida depois de alguns testes;
- Open track, utilizado para testes abertos, qualquer pessoa pode achar e baixar o aplicativo, serve para testes em massa e possui um aviso que o aplicativo ainda está em desenvolvimento. Pode demorar algumas horas para ser lançado;
- Production track, lançamento oficial aberto ao público. Pode demorar de horas a dias para ser lançado depois de testes e validações pela Google.

No caso do aplicativo de estudo foi usado o Open Track pela fácil distribuição.

3.3 Requisitos Técnicos

O aplicativo foi desenvolvido para o sistema operacional *Android* com a versão igual ou superior a 6.0 (Marshmallow, API 23) desta maneira foi possível abranger mais de dois terços do atuais usuários do sistema e garantir que o aplicativo seguisse as regras propostas pelo *Material Design*, que sugere a melhor forma de desenvolvimento para experiência de usuário. E com isso foi possível, por exemplo, utilizar *constraint layouts* em todas as telas, adaptando o aplicativo as diferentes dimensões de cada tela de celular sem perder informações ou apresentar falhas.

3.4 Requisitos de Funcionalidades

Um aplicativo que trata de um assunto abrangente precisa conseguir concentrar seus esforços em uma área de atuação, logística por si só é um tema que pode conter diversos estudos e logística terrestre em um país de vasto território como o Brasil pode trazer anos e anos de estudos e análises, por isso ao selecionarmos um ambiente específico podemos ter uma solução mais eficaz para o pontual que também que ajude o todo.

Para qualquer aplicativo temos dois requisitos básicos um deles é o público alvo, pois sem usuários um programa é totalmente inútil e sem sentido de existir. E outro requisito são suas funcionalidades, de nada adianta entregar um aplicativo que funciona, mas não atende os anseios e necessidades dos usuários.

No caso desse estudo o problema apareceu primeiro, que foi o motivador de que se desenvolvesse um aplicativo para logística de transporte terrestres, mas antes de qualquer desenvolvimento foram feitas entrevistas em estilo de conversas com o público alvo, procurando assim garantir um aplicativo que atende-se uma necessidade real.

3.4.1 Público Alvo

A seleção do público alvo se concentrou em dois perfis parecidos em ambos os casos seria necessário o trabalho com logística terrestre, mas ainda dentro desse ambiente fizemos entrevistas com pessoas de administração e engenharia de produção, buscando uma possível diversidade de respostas.

O que obtido por outro lado não atendeu as expectativas em questão de diversidade em relação ao curso. A diversidade das visões se mostrou presente mais em relação ao cargo ocupado do que em relação a teoria aprendida.

Por uma questão de tempo, contatos e distância física só foi possível analisar pessoas de uma mesma empresa, Votorantim Cimentos, porém dado o tamanho da empresa o material coletado se mostrou extremamente válido, separando o grupo de analise em dois grandes grupos um de Logística de Transporte de Produtos e outro grupo de Logística de Atendimentos.

Grupo de Transporte de Produtos

Nesse grupo estão presente os funcionários que se preocupam com a eficiência e custo do transporte. Durante as entrevistas foi identificado que o principal problema tratado é relacionado a dificuldade de planejamento ao se montar rotas entre os parques industriais e os depósitos ou entre depósitos e revendedoras/lojas/cliente.

O problema de uma plano cartesiano de complexidade N*M, onde os usuários precisavam de listas com n pontos e de uma fácil análise de conexão entre esses pontos. Principalmente pelo maior ponto de reclamação que era a situação quando um novo ponto era inserido e tudo precisava ser recalculado.

Desse grupo porém surgiu uma necessidade não pensada antes, ao se pensar na primeira versão da solução o foco era apenas a distância e foi durante essa entrevista que se percebeu a necessidade de também disponibilizar o tempo. Primeiro por uma questão de organização da empresa, de saber qual o tempo que a carga levará para completar o trajeto. E segundo, por pedido desses usuários, que fosse possível analisar o tempo de acordo com o trânsito e ou possíveis problemas na pista. Onde se mostrou a necessidade pela primeira vez de usar uma ferramenta com uma grande base de dados sobre o assunto, a *API Google Maps Platform*

Grupo de Logística de Atendimento

O segundo grupo comporta um trabalho mais dinâmico com solução de problemas diários, nesse grupo está a logística dos agentes em campo, desde os vendedores, aos atendentes de clientes e possíveis técnicos.

Nesse grupo surgiu uma necessidade nova pois existe uma grande perda de tempo do funcionário quando o planejamento diário dele sofre uma mudança. Se o agente externo se programa para passar por 10 lugares durante o dia e um ou mais desses lugares sofre uma mudança não podendo atender o agente na hora planejada isso gera um efeito cascata.

O funcionário liga para a central pedindo orientações, alguém da central precisa parar o que estava fazendo e focar nesse problema, muitas vezes o cliente quer remarcar para o mesmo dia, mais cedo ou mais tarde. Porém por questão logística essa reorganização pode levar um tempo precioso e muitas vezes é melhor deixar o agente em espera, sem efetuar

trabalho real, no horário do cliente que cancelou. Causando perda de tempo, necessidade de que remarcasse e atrasos.

Diante dessa situação a sugestão válida foi para que o agente externo possa usar sua localização atual como ponto de análise do aplicativo e que isso seja feito de forma remota, reforçando a ideia do aplicativo móvel.

Dessas entrevistas foi retiradas a funcionalidade de 1 para n nas listas, utilizando apenas sua localização atual para diversos endereços.

Ainda em conversas posteriores se descobriu mais um benefício dessa funcionalidade onde o agente externo quando está em um bairro fora do planejamento inicial pode usar a lista desse bairro para decidir por onde começar seu trajeto de forma rápida e eficiente, essa necessidade se dá algumas vezes quando o sistema de uma região cai impossibilitante o agente de agir em seu planejamento original.

3.4.2 Funcionalidades

Nessa seção poderemos aprofundar em cada funcionalidade do aplicativo. Focamos aqui em funcionalidades para seu funcionamento básico, atendendo o público alvo e sendo um programa válido de estudo e pesquisa. Diversas outras funcionalidades apareceram durante o desenvolvimento e testes finais, porém essas serão citadas na seção de Trabalhos Futuros 6.2 do Capítulo de conclusão pois apesar de extremamente válidas para solução de problemas ou melhor experiência do usuário um estudo deve focar um ambiente específico para ser eficaz, como comentado no começo deste capítulo.

Listas de Endereço

A base do aplicativo se dá nas listas de endereços, para que possamos garantir a veracidade dos endereços foi usada a API do Google Maps como ferramenta de pesquisa dos endereços e para pegar suas informações.

Além disso as listas podem ser criadas, nomeadas para identificação, conter diversos endereços, remover todos os endereços com um botão e adicionar e ou remover um endereço por vez

Localização

A localização do usuário se utiliza do GPS do próprio celular, dos pontos de conexão wifi e então busca os detalhes do endereço pela API do Google Maps, validando o resultado

Resultado

O resultado calcula todas as conexões entre os endereços de duas listas ou entre os endereços de uma lista e localização do aparelho. É possível apenas a seleção de duas listas ou uma lista e a localização do usuário por vez.

Algumas vezes o usuário pode inserir uma localização que não seja possível alcançar ou que não traga informações necessárias para rotas, quando isso acontece gera uma "não resposta"que pode ser filtrada na tela de resultados

No resultado podemos observar os endereços usados e sua distância em quilômetros e seu tempo de trajeto.

Abrir rota

Com o resultado em tela, caso o item seja selecionado a rota descrita entre os dois pontos é aberta no Google Maps para navegação direta do usuário.

Uso da memória do celular

Para o uso do aplicativo não é necessário autentificação, com isso se fez necessário que seja possível salvar listas diretamente na memória do celular. Para essa funcionalidade são permitidas apenas duas listas. Sendo então que sem autenticar o usuário pode ter 3 itens, duas listas e sua localização.

Salvar edição não salva

A criação de listas pode ser um trabalho extenso e demorado, uma preocupação percebida dos usuários é o retrabalho feito muitas vezes. Buscando melhorar essa situação foi implementada uma lista temporária que caso o trabalho e endereços das listas não seja salvo e seja interrompido por uma ligação, a bateria ter acabado ou algum motivo externo que faça com que o usuário bloquei o celular a lista está salva.

Ao entrar de novo para editar ou criar listas um diálogo verifica com o usuário se ele deseja continuar a edição da lista anterior não salva.

Autenticar

A autenticação do usuário é útil para resolver dois problemas, um deles em questão de portabilidade, caso se troque de celular é interessante que ainda se tenha acesso as listas criadas. E a questão de quantidade de listas salvas. Utilizando o banco de dados em nuvem passa-se de duas possíveis listas para um infinidade de listas.

Além de muitas outras funcionalidades que serão citadas nos trabalhos futuros.

Continuidade de uso

Após autenticar no aparelho é feita uma checagem para ver se existem listas salvas na memória do celular, se sim elas são transferidas para o banco de dados do aplicativo.

3.5 Arquitetura

Como o autor tem planos futuros que esse aplicativo possa ser disponibilizado como um projeto *Open-Source* para que evolua e possa ter ainda mais funcionalidades o principal motivo da escolha de uma arquitetura se deu por conta da manutenção, substituição e alteração do código com facilidade.

Vale citar a importância de arquitetura para outros elementos como redução de custos [19], gerenciamento de riscos [20] e o desempenho do código [21].

Para esse programa foi usado MVC (*Model-View-Controller*) separando assim as informações e classes de interface, data e métodos. Como exposto na Figura 3.1

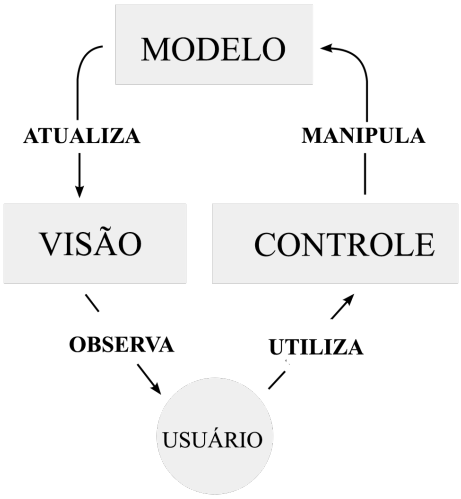


Figura 3.1: Diagrama de interações com o padrão MVC. Tradução livre (Fonte: [22]).

Temos então uma situação delicada, pois o uso de arquitetura com metodologia Ágil é algo estranho de se pensar visto que o primeiro visto fixar o formato do desenvolvimento desde o início até o final e o segundo propõe que o formato vai crescendo e se adaptando de acordo com o desenvolvimento. E realmente existem problemas estruturais de se pensar em uma arquitetura completa do início ao fim de forma um pouco mais rígida para uma metodologia tão mutável e adaptável ao problema diariamente, mas para esse problema foram desenvolvidos algum métodos com foco nessa situação como a recuperação. [23]

Capítulo 4

O Aplicativo

O aplicativo busca auxiliar a logística de transporte terrestres em dois momentos um durante o planejamento que pode ser considerado um trabalho dentro do escritório e outro durante a execução que poderíamos considerar como um trabalho em campo.

A premissa do aplicativo é de ser uma ferramenta simples e que ao mesmo tempo resolva os problemas, sendo assim buscou-se evitar criar redundâncias como um menu para navegação, deixando então a função de voltar para a tela anterior a cargo do botão de navegação nativo do sistema *Android*. Com isso também ganhou-se espaço em tela para visualizar as listas, que podem crescer de forma rápida e em multiplicidade a cada endereço incrementado.

A linha lógica de uso do aplicativo tem dois caminhos, um deles que representa apenas a autentificação do usuário e depois suas informações com opção de desvincular a conta no aplicativo e sua principal função, qual consiste em ver as listas, criar, deletar e editar listas, selecionar listas para gerar os resultados de distância e exportar esses resultados para navegação no *Google Maps*.

Nas secções seguintes serão detalhadas as telas mais importantes do aplicativo.

4.1 Tela Inicial

Visto que o aplicativo foi criado para validação e teste da ideia a tela inicial (Figura 4.1) foi desenvolvida com um foco de explicação dos possíveis uso do aplicativo e apresenta as duas opções iniciais

Nela encontramos o texto em sua versão em português e dois botões:

- Autenticação/Perfil: Autenticar, ver perfil e sair.
- Resultados: Encaminha para as listas de endereços.

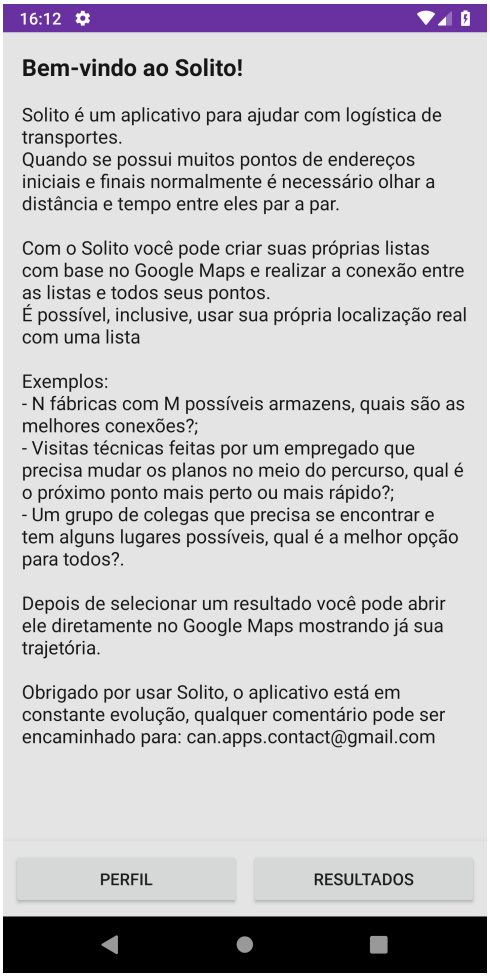


Figura 4.1: Tela inicial.

4.2 Autenticação

Na tela de autenticação (Figura 4.2) o usuário pode criar sua conta apenas informando o endereço eletrônico e a senha desejada, usando os mesmos campos ele pode logar em sua conta ou pode se utilizar da sua conta *Google* para acessar o aplicativo.

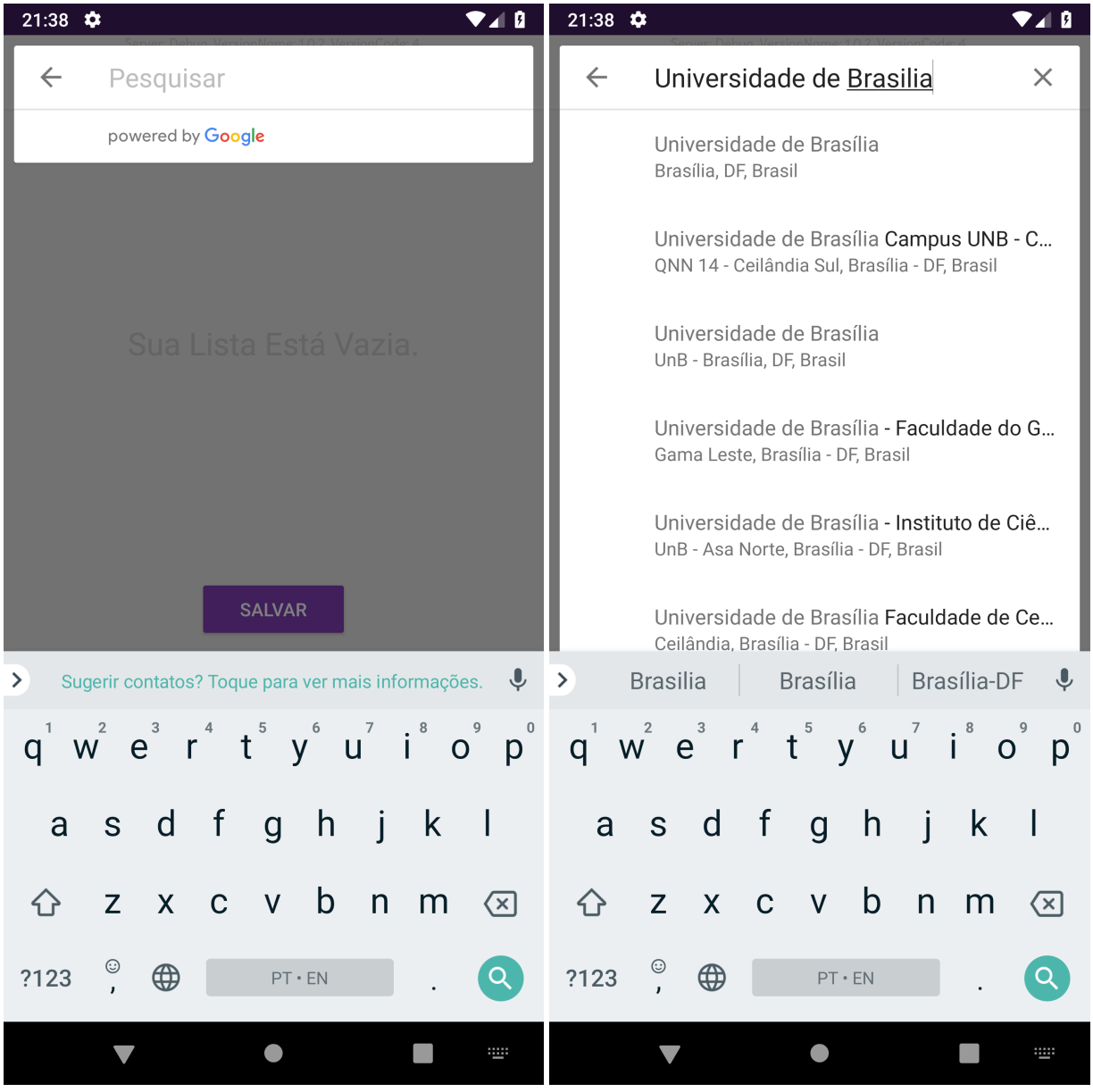
Para o uso do aplicativo a autenticação não é necessária, porém após entrar em sua conta o usuário tem mais recursos. Antes o usuário estava restrito a apenas duas listas salvas no celular, autenticado as listas do celular são salvas automaticamente em sua conta e ficam salvas também no banco de dados, podendo ser acessadas de qualquer aparelho. Outro recurso é a criação de um número maior de listas de endereços e pode manter-se conectado no mesmo aplicativo, facilitando o uso sem precisar se reconectar toda vez que for usar a ferramenta.

Figura 4.2: Tela de autenticação e cadastro.

4.3 Busca

As telas de busca são acessadas quando se cria ou edita uma lista. Um importante ponto do aplicativo é garantir que os endereços selecionados são de locais reais, assim o uso da *API* do *Google Maps* (Figura 4.3a) se tornou uma característica importante da ferramenta onde somente endereços reconhecidos pela ferramenta são retornados (Figura 4.3b), trazendo uma maior garantia de possível análise para o caminho.

Importante ressaltar que o uso de endereços muito abrangentes, como por exemplo 'Brasília', pode ocasionar em resultados vazios, já que a interpretação do endereço pode não retornar um ponto específico.



(a) Busca fornecida pela Google. (b) Lista de resultados

Figura 4.3: Tela de busca.

4.4 Listas

As listas do aplicativo são sua funcionalidades principal, começando apenas com a opção de utilizar sua própria localização a tela de listas impede que o usuário passe adiante sem selecionar dois itens como observado na (Figura 4.4). O botão de calcular resultados está desativado.

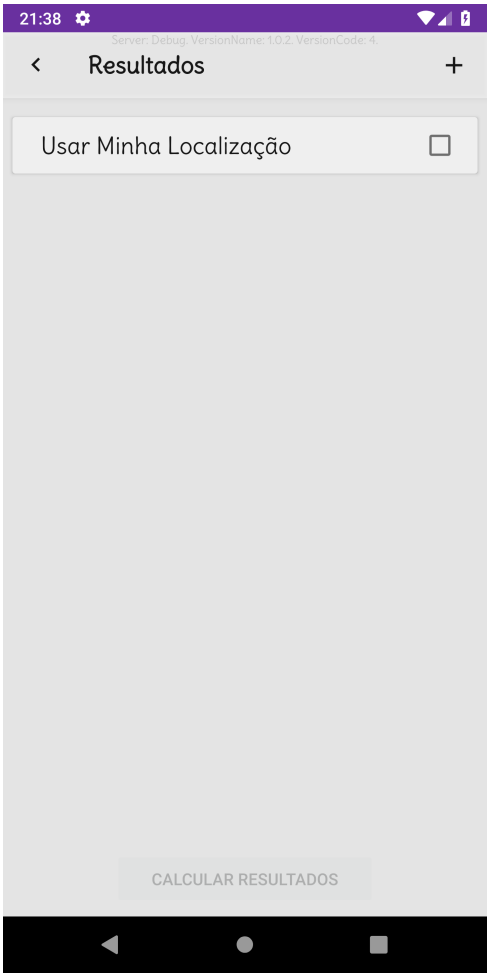


Figura 4.4: Tela de Listas Vazia..

A tela possui um botão de somatório no canto superior direito que fornece a opção de criar uma lista.

4.4.1 Criando uma lista de endereços

O primeiro passo para a criação de listas é fornecer um nome por meio de um diálogo que vemos na Figura 4.5.

Após fornecer o nome da lista a tela dela é apresentada vazia (Figura 4.6a) até que busquemos os endereços e a preenchermos (Figura 4.6b).

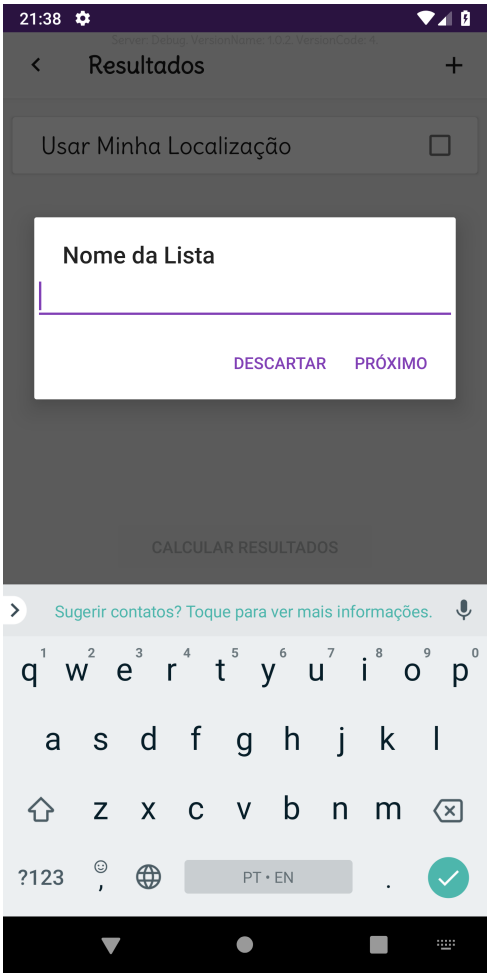
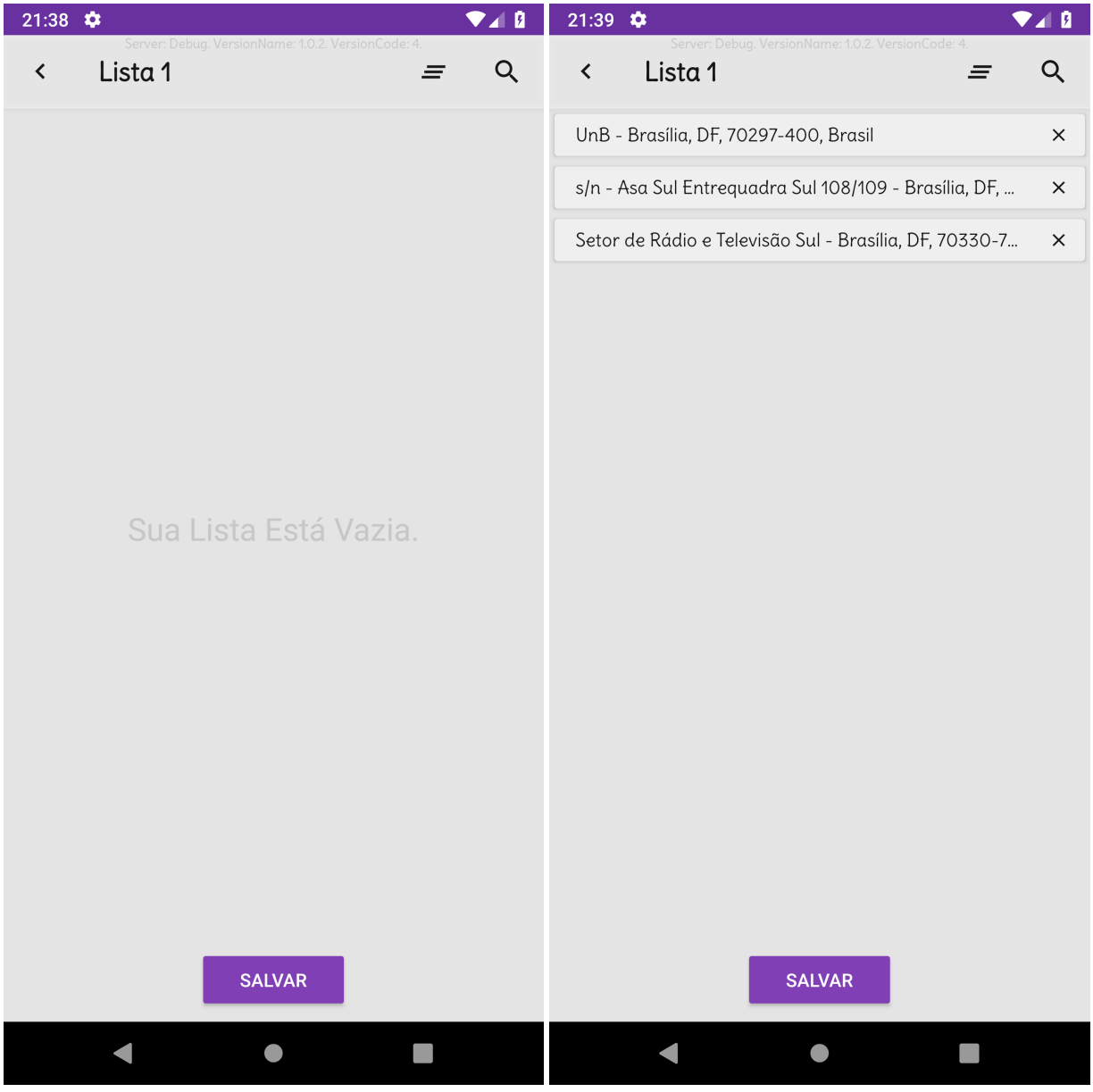


Figura 4.5: Tela com diálogo para dar nome a lista..

As buscas nesse caso são salvas localmente no celular até que sejam salvas na lista, assim evitamos perder o trabalho de seleção de endereços caso o telefone toque, o aplicativo feche ou a bateria acabe. A última lista editada sempre aparecerá como opção para o usuário caso ele deseje continuar editando sem perder seu trabalho.

Nessas telas nos deparemos com os botões:

- Pesquisar: que abre as telas da busca, seção 4.3.
- Três Barras em diagonal: Apesar de pouco conhecido o ícone, baseado no *Material Design* da *Google* representa apagar todos os elementos.
- Salvar: Salva a lista localmente ou na base de dados e evita salvar uma lista vazia.
- Botão X: Apaga individualmente cada endereço da lista.



(a) Lista vazia. (b) Lista preenchida.

Figura 4.6: Tela de edição para listas.

4.5 Resultados

Após a criação das listas é possível selecionar duas delas (Figura 4.7) (ou umas delas e sua localização) e gerar resultados que informam distância e tempo previsto.

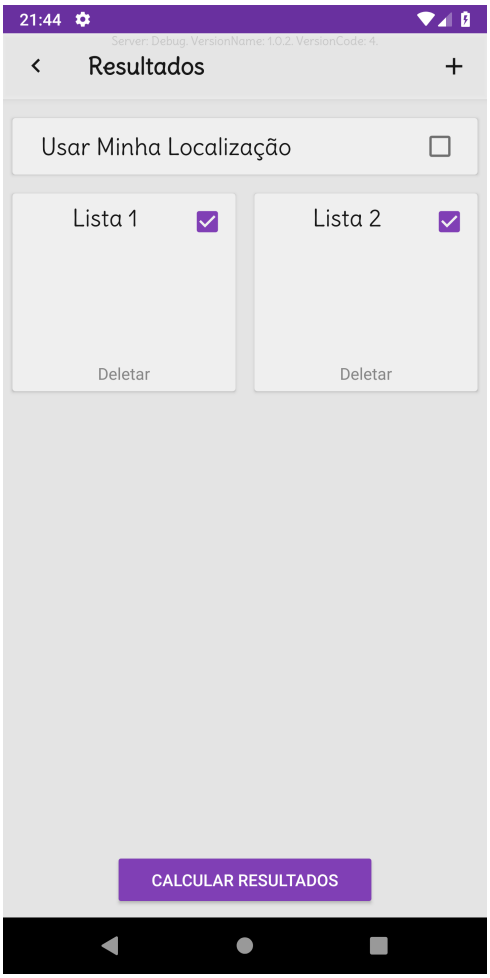
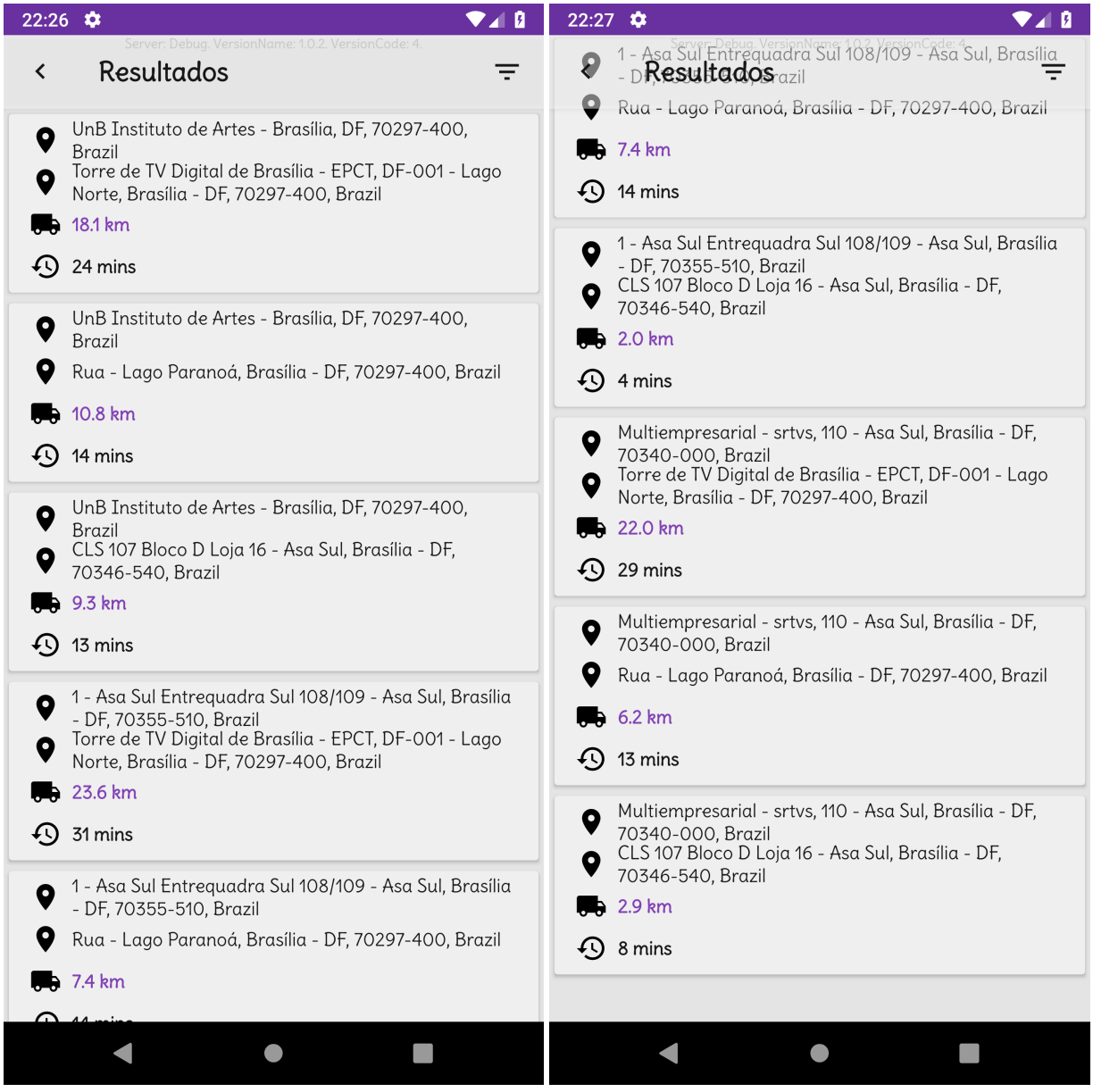


Figura 4.7: Tela com listas selecionadas para calcular resultado.

Com as listas de endereços foi possível utilizar a Google API para enviar blocos de 24 endereços por vez retornando os valores de tempo e distância entre eles. Assim geramos todos os caminhos possíveis entre os endereços das listas como demonstrado na Figura 4.8

Nesta tela podemos clicar no botão de filtro que retira da visualização os resultados que voltaram vazios, por exemplo se é necessário utilizar de transporte marítimo. E clicando na caixa do resultado abre-se o trajeto no *Google Maps*.



(a) (b)

Figura 4.8: Tela de resultados com cartões de tempo e distância.

Capítulo 5

Validação e Testes

O teste de aplicativos é uma função de extrema importância que deve ser executada por profissionais treinados e de preferência com experiência. Infelizmente esse grupo de testadores não se contra disponível para essa pesquisa, sendo assim na falta de especialistas foi montado um plano para suprir essa necessidade.

Foi enviado um convite para diversos possíveis usuários, o número de interessados em testar o aplicativo foi de 17. Para esse grupo foi criado um tempo de uso 5.1 e uma metodologia diária de notificações 5.2 buscando evitar a exaustão nos testes.

Ao final do desenvolvimento e testes do aplicativo ficou claro o quanto a suposição inicial se confirmou durante o processo. Existe uma defasagem em aplicativos para melhores soluções no campo de logística terrestre. Ainda foi de grande valia o *feedback* recebido sobre o aplicativo em si e a empolgação dos usuários, mesmo antes de usar o produto, quando entendiam do que se tratava o problema, desde casos mais simples, como achar o melhor ponto de encontro para um grupo de amigos, até casos mais complexos envolvendo rotas de transportadoras a usabilidade. A solução proposta é claramente um desejo dos envolvidos nos testes. Isso não confirma sua aprovação imediata no mercado de trabalho logístico, porém já proporciona fortes indícios de uma aceitação inicial.

Neste capítulo focaremos sobre a metodologia para validação e testes do produto. Tivemos 3 pontos principais que influenciam a coleta de dados o tempo de uso: o público que foi selecionado como amostra e quais resultados as interações deles tiveram baseado em um questionário aplicado após alguns dias de uso.

5.1 Tempo de Uso

Para recolher uma melhor opinião dos usuários, durante os testes, foi decidido que apenas uma interação com o aplicativo poderia resultar em uma análise incompleta. Objetivo

então era fazer com que os usuários tivessem mais interações. Para isso foi fixado um espaço de tempo de 5 dias, com uma interação cada dia.

Os 5 dias surgiram após perceber que testar todas as funcionalidades em sequência seria extenso e massante para o usuário *beta*. Sendo assim dividimos as principais funcionalidades uma em cada dia, sendo elas:

- Criação de listas com Busca de endereços;
- Edição e exclusão de listas;
- Gerar resultados e abrir navegação externa;
- Utilizar a localização atual;
- Autenticar no aplicativo.

Para incentivar esse uso específico das funcionalidades durante os dias foi desenvolvido por meio do *Firebase* um sistema de notificações no celular.

5.2 Notificações

Usando um painel administrativo foi planejado e enviado lembretes diários de uso, em horários diferentes para cada dia.

Durante a explicação do aplicativo foi enfaticamente debatida a importância do uso diário para o público teste. Excluindo o primeiro dia de download e uso tivemos 5 dias de mensagem para 17 usuários, totalizando 85 notificações.

Pudemos observar um bom resultado nessa estratégia onde das 85 notificações foram abertas 71. Um aproveitamento de 84%. Cada uma das notificações continham sugestões de uso, como:

As mensagens das notificações sempre tinham o foco de indicar qual interação o usuário deveria executar.

- Sugestão do dia: Ao criar listas verifique se a busca está com valores reais;
- Sugestão: Como está funcionando a edição e exclusão de listas?;
- Selecionando duas listas o resultado é gerado, já tentou pressionar ele?;
- E se você utilizar sua localização atual com uma lista de endereços?;
- Você sabia? Autenticando no aplicativo você consegue criar mais listas.

5.3 Público Teste

O aplicativo foi testado com 3 grupos diferentes de usuários. Dois dos grupos focados no usuário final e um dos grupos focado em desenvolvedores para analisar a consistência do aplicativo.

Inicialmente foram 20 selecionados, infelizmente essa seleção foi feita no início das atividades da pesquisa o que acarretou uma complicação na hora avaliação do aplicativo. Muitos dos interessados, principalmente do público alvo, não possuíam o sistema operacional *Android* em seus celulares.

Sendo assim, após substituições foram 17 avaliadores durante um período de 5 dias de uso (com lembretes diários pelo próprio aplicativo) e ao final os avaliadores responderam um questionário online.

Foram 6 desenvolvedores, 7 estudantes ou formados da área de Administração focada em Logística e 4 estudantes ou formados em Engenharia de Produção que também enfocam a logística. Desse grupo é interessante destacar que 6 usaram o aplicativo em inglês por não falarem português. Do grupo de desenvolvedores 4 deles são programadores para *mobile* (*iOS*ou *Android*).

Dos grupos focados no usuário final (Administradores e Engenheiros de Produção) 7 deles estavam trabalhando com logística durante os testes, como estagiários ou efetivos.

5.4 Questionário

Após o período de testes foi aplicado um questionário para os usuários. Nossa metodologia então foi de ter testes contantes do usuários e ao final fornecer o formulário com perguntas específicas das funcionalidades.

A Tabela 5.1 contem o resumo das respostas, pode-se observar que nem todos as respostas foram respondidas por todos os participantes. A tabela contem a pergunta, sua resposta positiva, negativa e um meio termo. Porém algumas perguntas são de resposta binária. E por isso apresentam campos com hífen.

Uma das questões sobre utilidade para logística de transportes não foi respondida por nenhum desenvolvedor, mesmo que nenhuma instrução tenha sido fornecida nessa direção.

A idade média dos avaliados foi de 25 anos. Com mínima de 23 e máxima de 29.

Foi aberto um espaço para comentários ao final, além dos comentários positivos e falando como o aplicativo parece promissor, um dos comentários, traduzido do inglês, tem um peso maior para o desenvolvimento do aplicativo.

"O aplicativo pode ser usado de forma útil, mas primeiro foi difícil entender como ele funcionava. Se tiver mais explicações com certeza seria melhor. E comparado com outros

Tabela 5.1: Respostas coletadas após o uso do aplicativo (em porcentagem).

Resultados Solito			
Pergunta	Sim	+-	Não
O uso do aplicativo é intuitivo?	24	41	35
Criação de listas funcionou?	82	18	0
Selecionar localização atual funcionou?	100	0	0
Selecionar itens funcionou adequadamente?	88	6	6
Selecionar endereços encontrou os endereços certos?	64	18	18
As listas foram salvas no seu celular?	76	24	0
Os resultados foram satisfatórios?	71	17	12
Os resultados abriram no Google Maps?	88	12	0
Foi possível nomear as listas?	94	6	0
O registro e login funcionou?	71	29	0
Após login, foi possível criar mais de duas listas?	94	0	6
As listas criadas anteriormente ao login, foram salvas após o login?	100	0	0
A ordenação das listas funcionou?	82	6	12
A interface do aplicativo é amigável?	88	0	12
A linguagem do aplicativo é adequada?	65	12	23
O aplicativo responde de forma rápida aos comandos?	100	0	0
As cores do aplicativo são agradáveis?	76	12	12
Os ícones ou comandos são explicados claramente?	76	12	12
A diagramação da tela é adequada quanto à colocação de textos?	100	0	0
Os ícones utilizados são compreensivas para sua função?	76	18	6
A sequência de apresentação das telas favorece a compreensão do conteúdo?	82	6	12
O programa resiste a respostas inadequadas sem travar (como, por exemplo, digitar teclas erradas)?	88	12	0
As telas são de fácil leitura?	76	24	0
Os textos apresentados nas telas são claros?	100	0	0
As mensagens de erro são explicativas?	82	18	0
O programa é “leve” e fácil de ser executado pelo celular?	100	0	0
O aplicativo está em condição satisfatória para uso?	76	-	24
O aplicativo tem sua utilidade para logística de transportes?	65	-	0
Conhece algum outro aplicativo (gratuito ou não) que execute funções similares?	0	-	100
Você é parente ou familiar ou mesmo possui relação de amizade com o autor deste aplicativo?	65	-	35

aplicativos que temos hoje no mercado(não relacionados a navegação) ele é maçante, poderia ser mais colorido e mais compreensível, sabendo que as pessoas baixam aplicativos para fazer suas vidas mais simples e que assim que baixam os usuários querem a informação que eles precisam de forma fácil, eles não tentariam entender e descobrir como funciona.- Özge Nur Zor, tradução livre.

Comentário feito por um desenvolvedor de *iOS*, nascido na Turquia e hoje trabalhando em uma fábrica de softwares em Portugal. Traz para pesquisa uma análise importante sobre o peso da Experiência de Usuário nas interações.

Capítulo 6

Conclusão

Este trabalho teve como foco o desenvolvimento de um aplicativo para celular (com sistema *Android*) para facilitar o dia a dia de logística terrestre em relação aos caminhos possíveis. Começou apenas para resolver um problema de diversos pontos de partidas com diversos pontos de chegada e fazer essa análise de forma automática para se encontrar a melhor solução. Porém, durante o trabalho percebeu-se um leque muito maior de atuação dessa ferramenta, pois a utilidade dela não seria apenas pontual na hora do planejamento, mas também funcionaria com maior dinamicidade podendo reajustar as possíveis rotas ou ainda tomar a localização atual do usuário como ponto de partida.

Foi realizado levantamento de requisitos junto a profissionais atuantes na área de logística de transporte, em busca de serem identificadas necessidades de soluções para o dia a dia. As ferramentas de desenvolvimento envolveram o Android Studio e o Firebase; utilizou-se a abordagem de métodos ágeis. O aplicativo desenvolvido contempla funcionalidades importantes para essa área, pois consegue salvar listas de locais de forma limitada, caso não esteja logado, mas ilimitada com usuários logados, e mantém essas listas salvas no database para garantir mobilidade.

Ainda, utiliza *Google API* para melhor certeza dos endereços buscados e utilizados, consegue usar a localização real e atual do usuário, bem como exporta o trajeto escolhido para navegação. Durante a fase de desenvolvimento, a apresentação do projeto para o público alvo e conversas sobre o desenvolvimento do mesmo causaram alto nível de curiosidade e de animação. Isto porque o problema que o aplicativo busca resolver é relevante.

Foi possível auxiliar no planejamento de rotas para outras pesquisas e incentivar o uso de listas em busca de soluções ótimas. Testes realizados por diversos profissionais da área de logística em transporte apontaram que essa ferramenta possui características de funcionalidades úteis para esse setor. Foi possível apresentar para o usuário as medições de tempo e distância fornecidas pela ferramenta.

6.1 Mudança no Google Maps

Uma ação da *Google* que ocorreu durante o desenvolvimento da aplicação pode ser analisada para validar, ainda mais, a ideia da logística por navegação com transportes terrestres.

Durante as fases iniciais do aplicativo o uso da *Google Maps API* era de forma gratuita utilizando uma chave que estava ligada a sua conta de desenvolvedor.

Após um certo número de utilizações o sistema impedia mais requisições utilizando a mesma chave.

Esse modelo de negócio no entanto sofreu duas mudanças, primeiro ele passou a fornecer um crédito de 200 dólares para o desenvolvedor e até alcançar esse valor a utilização era normal, para continuar utilizando após isso era necessário inserir crédito na conta ou fornecer um número de cartão de crédito para débito automático baseado na utilização.

E posteriormente outra mudança passou a impedir a utilização do crédito caso a conta não contenha um cartão de crédito válido.

Claro que esse modelo de negócio potencializa os ganhos da empresa no sentido que não trava o funcionamento da ferramenta e a cada uso acima do crédito mensal gera um débito. Porém também ajuda a analisar o uso das *Google Maps APIs*, com essa mudança de modelo de negócio observamos que mais pessoas mundialmente estão utilizando essas ferramentas, tornando a navegação terrestre uma tendência em ascensão.

6.2 Trabalhos Futuros

E durante o desenvolvimento, as pesquisas e os testes finais diversas novas ideias surgiram que podem incrementar sua funcionalidade e ajudar na logística terrestre, desde ideias para melhor experiência do usuário como ideias de novas funções para soluções de problemas

1. Migrar o sistema para Open Street Maps
2. Copiar listas;
3. Compartilhar entre usuários;
4. Análise de rota em tempo futuro, prevendo o a média de trânsito do dia seguinte, por exemplo;
5. Filtrar respostas por cidade, listas, velocidade ou distância;
6. Achar ponto de encontro ótimo (melhor ponto médio na lista B para todos os pontos na lista A);

7. Utilizar mais meios de transportes, bicicleta, transporte público, etc.

Espera-se que colocando esse aplicativo em código aberto ela possa ser incrementado por futuros estudantes e pesquisadores até que se chegue em um ponto onde grandes e médias companhias possam utiliza-lo melhorando o tráfico de produtos e bens como até ativando outros pontos de coleta, armazenamento ou distribuição que não são tão ativos por falta de conhecimento de sua melhor eficácia.

6.3 Possíveis Problemas

Após os testes executados e questionários respondidos, durante a finalização desta monografia a *Google Store* enviou uma notificação sobre a remoção do aplicativo de sua loja por uma "Violação de uso da Identidade do Android [...]"que está ligada a identificação do usuário e seus dados.

Seria então parte do processo criar um Certificado de Privacidade a ser aceito pelo usuário, para o uso de sua posição global (GPS), por exemplo. Dado que a segurança de dados vem se tornando cada vez mais um assunto discutido e delicado é importante entender como o usuário final se sentirá tendo sua trajetória gravada diversa vezes ao dia.

O comércio de vender a localização das pessoas é gigante [24], vale-se então de deixar claro para os usuários os fins acadêmicos e altruístas da aplicação.

Referências

[1] M. P. Hamaoka, "Análise de potencial de demanda para exportação pelo porto do itaqui," *Monografia Universidade de Brasília*, p. 10, 2018. 1

[2] Google, "Sítio de navegação google maps." `https://www.google.com/maps`. Acesso em: 15 de Dezembro de 2018. 2

[3] StatCounter, "Operating system market share worldwide." `http://gs.statcounter.com/os-market-share#monthly-201706-201806-bar`. Acesso em: 22 de Novembro de 2018. 5

[4] R. H. Ballou, *Logística empresarial: transportes, administração de materiais e distribuição física*. Atlas, 2011. 6

[5] D. J. C. Donald J Bowersoz, *Logística empresarial: o processo de integração da cadeia de suprimento*. Atlas, 2001. 6

[6] P. L. de Mesquita, "Sistema portuário nacional." `http://www.portosdobrasil.gov.br/assuntos-1/sistema-portuario-nacional`. Acesso em: 07 de Setembro de 2018. 6

[7] G1, "Industrias brasileiras." `http://g1.globo.com/economia/noticia/2013/06/brasil-tem-312-mil-industrias-com-um-ou-mais-funcionarios-diz-ibge.html`. Acesso em: 07 de Setembro de 2018. 6

[8] IBID, "O cenário de armazenamento e logística no brasil." `https://ibid.com.br/blog/o-cenario-de-armazenagem-e-logistica-no-brasil/`. Acesso em: 07 de Setembro de 2018. 6

[9] Conab, "Companhia nacional de abastecimento." `https://www.conab.gov.br`. Acesso em: 07 de Setembro de 2018. 6

[10] M. Brown, "The "lost" steve jobs speech from 1983." `http://lifelibertytech.com/2012/10/02/the-lost-steve-jobs-speech-from-1983-foreshadowing-wireless-networking` Acesso em: 22 de Novembro de 2018. 7

[11] C. Williams, "App stores start to mature – 2016 year in review." `https://blog.appfigures.com/app-stores-start-to-mature-2016-year-in-review/`. Acesso em: 22 de Novembro de 2018. 7

[12] M. Strain, "1983 to today: a history of mobile apps." https://www.theguardian.com/media-network/2015/feb/13/history-mobile-apps-future-interactive-timeline. Acesso em: 23 de Novembro de 2018. 7

[13] D. Kushner, "The flight of the birdman: Flappy bird creator dong nguyen speaks out." https://www.rollingstone.com/culture/culture-news/the-flight-of-the-birdman-flappy-bird-creator-dong-nguyen-speaks-out-112457/. Acesso em: 22 de Novembro de 2018. 8

[14] B. News, "Bbc launches whatsapp ebola service." https://www.bbc.co.uk/news/world-africa-29573964. Acesso em: 22 de Novembro de 2018. 8

[15] B. V. Boas, "Um em cada 4 brasileiros de dez anos ou mais não tem celular." https://www.valor.com.br/brasil/5337797/um-em-cada-4-brasileiros-de-dez-anos-ou-mais-nao-tem-celular-diz-ibge. Acesso em: 02 de Dezembro de 2018. 8

[16] C. for Medicare & Medicaid Services (CMS), "Selecting a development approach.," *Office of Information Servic*, p. 10, 2008. 9

[17] S. Ambler, "Examining the agile manifesto." http://www.ambysoft.com/essays/agileManifesto.html. Acesso em: 17 de Setembro de 2018. 9

[18] J. Highsmith, "History: The agile manifesto." http://agilemanifesto.org/history.html. Acesso em: 17 de Setembro de 2018. 9

[19] E. Poort, "Rcda: Architecting as a risk and cost management discipline," *The Journal of Systems and Software*, vol. 85, no. 9, 2012. 17

[20] G. Fairbanks, *Just Enough Software Architecture*. Marshall & Brainerd, 2010. 17

[21] J. Bosh, *Design and use of software architectures*. Addison-Wesley, Harlow, 2000. 17

[22] T. Reenskaug and J. Coplien, "The dci architecture: A new vision of object-oriented programming." http://www.artima.com/articles/dci_vision.html. Acesso em: 22 de Novembro de 2018. 17

[23] B. B. . R. Turner, *Balancing Agility and Discipline*. Addison-Wesley, 2004. 17

[24] M. Barbaro, "The business of selling your location." https://www.nytimes.com/2018/12/10/podcasts/the-daily/location-tracking-apps-privacy.html?action=click&module=audio-series-bar&region=header&pgtype=Article. Acesso em: 22 de Novembro de 2018. 34

Anexo I

Código fonte Solicito

I.1 Telas

I.1.1 Autentificação

```
1
2  class AuthenticationActivity : BaseActivity(), GoogleApiClient.
       OnConnectionFailedListener {
3
4      //Authentication
5      private var mAuth: FirebaseAuth? = null
6      //Google auth
7      private lateinit var gso: GoogleSignInOptions
8      private var mGoogleApiClient: GoogleApiClient? = null
9
10     //Not in Layout
11     private var passwordVisibility = false
12     private var validEmail = false
13     private var validPassword = false
14     private val SIGN_IN_GOOGLE = 369
15     private var firestoreDatabase: FirebaseFirestore =
           FirebaseFirestore.getInstance()
16
17     ////////////////////  IMPLEMENT METHODS  ////////////////////
18     override fun initializeActionBar() {
19         actionbarLeftBtn.visibility = View.VISIBLE
20         actionbarTitle.visibility = View.VISIBLE
21     }
```

```
22
23      override fun getActionbarTitle(): String {
24          return resources.getString(R.string.auth_activity)
25      }
26
27      override fun getContentView(): Int {
28          return R.layout.authentication_activity
29      }
30
31      override fun assignViews() {
32          mAuth = FirebaseAuth.getInstance()
33          gso = GoogleSignInOptions.Builder(GoogleSignInOptions.
                DEFAULT_SIGN_IN)
34                  .requestIdToken(getString(R.string.
                        default_web_client_id))
35                  .requestEmail()
36                  .build()
37          mGoogleApiClient = GoogleApiClient.Builder(this)
38                  .enableAutoManage(this, this)
39                  .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
40                  .build()
41      }
42
43      override fun prepareViews() {
44
45          sign_in_button_google.setSize(SignInButton.SIZE_WIDE)
46
47          sign_in_button_google.setOnClickListener{ view ->
48              if (Utils.isDoubleClick()) return@setOnClickListener
49              registerGoogle(view)
50          }
51
52          auth_email_text.addTextChangedListener(object :
                TextWatcher {
53              override fun beforeTextChanged(s: CharSequence?,
                    start: Int, count: Int, after: Int) {}
54              override fun onTextChanged(s: CharSequence?, start:
                    Int, count: Int, after: Int) {
55                  checkEmailFormat()
56              }
```

```
57              override fun afterTextChanged(s: Editable?) {}
58          })
59
60          auth_email_text.onFocusChangeListener = View.
               OnFocusChangeListener { _, hasFocus ->
61              if (!hasFocus) {
62                  checkEmailFormat()
63              }
64          }
65
66          auth_password_text.addTextChangedListener(object :
               TextWatcher {
67              override fun beforeTextChanged(s: CharSequence?,
                   start: Int, count: Int, after: Int) {}
68              override fun onTextChanged(s: CharSequence?, start:
                   Int, count: Int, after: Int) {
69                  if (auth_password_text.length() < 6) {
70                      auth_password_warning.visibility = View.
                           VISIBLE
71                      validPassword = false
72                  } else {
73                      auth_password_warning.visibility = View.
                           INVISIBLE
74                      validPassword = true
75                  }
76                  enableNextButton()
77              }
78              override fun afterTextChanged(s: Editable?) {}
79          })
80          auth_password_visibility_btn.setOnClickListener{
               passwordVisibility() }
81      }
82
83
84      ////////////////////  FUNCTIONS  ////////////////////
85      private fun passwordVisibility() {
86          if (passwordVisibility) {
87              auth_password_visibility_btn.setImageDrawable(
                   ContextCompat.getDrawable(this, R.drawable.
                   ic_visibility_off_black_24dp))
```

```
88              auth_password_text.inputType = InputType.
                    TYPE_CLASS_TEXT or InputType.
                    TYPE_TEXT_VARIATION_PASSWORD
89              passwordVisibility = false
90          } else {
91              auth_password_visibility_btn.setImageDrawable(
                    ContextCompat.getDrawable(this, R.drawable.
                    ic_visibility_black_24dp))
92              auth_password_text.inputType = InputType.
                    TYPE_CLASS_TEXT or InputType.
                    TYPE_TEXT_VARIATION_VISIBLE_PASSWORD
93              passwordVisibility = true
94
95          }
96      }
97
98      private fun checkEmailFormat() {
99          if (android.util.Patterns.EMAIL_ADDRESS.matcher(
                auth_email_text.text.toString().trim({ it <= ' ' })).
                matches()) {
100             auth_email_text.error = null
101             validEmail = true
102         } else {
103             auth_email_text.error = getString(R.string.
                    error_email_format)
104             validEmail = false
105         }
106         enableNextButton()
107     }
108
109     private fun checkUserLogin(){
110         val account = GoogleSignIn.getLastSignedInAccount(this)
111         if(account != null) {
112             firebaseAuthWithGoogle(account)
113         }
114         else{
115             val user = mAuth!!.currentUser
116             if (user != null) updateUI(user)
117         }
118
```

```
119
120     }
121
122     private fun enableNextButton() {
123         if (validEmail && validPassword) {
124             auth_get_in_btn.setBackgroundColor(ContextCompat.
                    getColor(this, R.color.solito_text_purple))
125             auth_get_in_btn.setOnClickListener{ view ->
126                 if (Utils.isDoubleClick())
                        return@setOnClickListener
127                 register(view)
128             }
129         } else {
130             auth_get_in_btn.setBackgroundColor(ContextCompat.
                    getColor(this, R.color.solito_button_grey))
131             auth_get_in_btn.setOnClickListener{ }
132         }
133     }
134
135     private fun showMessage(view: View, message: String){
136         Snackbar.make(view, message, Snackbar.LENGTH_LONG).
                setAction("Action", null).show()
137     }
138
139     ////////////////////  LISTS FUNCTIONS  ////////////////////
140     private fun checkPersistenceLists() {
141         if(DataStorePersistence.getInstance().getAddressListOne(
                baseContext).listID.isNotEmpty()){
142             saveDatabaseList(DataStorePersistence.getInstance().
                    getAddressListOne(baseContext))
143
144             DataStorePersistence.getInstance().
                    removeAddressListOne(baseContext)
145         }
146         if(DataStorePersistence.getInstance().getAddressListTwo(
                baseContext).listID.isNotEmpty()){
147             saveDatabaseList(DataStorePersistence.getInstance().
                    getAddressListTwo(baseContext))
148
```

```
149             DataStorePersistence.getInstance().
                    removeAddressListTwo(baseContext)
150         }
151     }
152
153     private fun saveDatabaseList(arrayList: JSONAddressList) {
154
155         //Create a new document
156         val addedDocRef: DocumentReference = firestoreDatabase.
                collection(Parameters.USER)
157                 .document(DataStorePersistence.getInstance().
                        getUserID(baseContext))
158                 .collection(Parameters.USER_LIST)
159                 .document()
160
161         //Get document ID
162         arrayList.listID = addedDocRef.id
163
164         //Update Document
165         addedDocRef.set(arrayList).addOnFailureListener { e ->
166             Crashlytics.log(e.message)
167             showErrorToast(baseContext, e.message)
168         }.addOnCompleteListener { task ->
169             if (task.exception != null) {
170                 showErrorToast(baseContext, task.exception.
                        toString())
171             }
172         }
173
174     }
175
176
177     ////////////////////  REGISTER  ////////////////////
178     private fun register(view: View){
179         val email = auth_email_text.text.toString()
180         val password = auth_password_text.text.toString()
181
182
183         showMessage(view,resources.getString(R.string.
                str_authentication_wait))
```

```
184
185         mAuth!!.signInWithEmailAndPassword(email, password)
186                 .addOnCompleteListener{ taskSign ->
187                     if(taskSign.isSuccessful){
188                         //Success Login
189                         updateUI(mAuth!!.currentUser)
190                         Toast.makeText(this, resources.getString(
                                R.string.str_sign_in_success), Toast.
                                LENGTH_SHORT).show()
191                     }else{
192                         mAuth!!.createUserWithEmailAndPassword(
                                email, password)
193                                 .addOnCompleteListener {
                                    taskCreate ->
194                                     if (taskCreate.isSuccessful)
                                        {
195                                         //Success Create User
196                                         updateUI(mAuth!!.
                                            currentUser)
197                                         Toast.makeText(this,
                                            resources.getString(R.
                                            string.
                                            str_create_success),
                                            Toast.LENGTH_SHORT).
                                            show()
198                                     } else {
199                                         //Fail Login
200                                         Toast.makeText(this,
                                            resources.getString(R.
                                            string.
                                            str_wrong_password),
                                            Toast.LENGTH_SHORT).
                                            show()
201                                     }
202                                 }
203                     }
204                 }
205
206     }
207
```

```
208     private fun registerGoogle(view: View){
209         showMessage(view,resources.getString(R.string.
                str_authentication_wait))
210         val signInIntent = Auth.GoogleSignInApi.getSignInIntent(
                mGoogleApiClient)
211         startActivityForResult(signInIntent, SIGN_IN_GOOGLE)
212     }
213     ////////////////////  REGISTER RESULTS  ////////////////////
214
215     override fun onActivityResult(requestCode: Int, resultCode:
            Int, data: Intent?) {
216         super.onActivityResult(requestCode, resultCode, data)
217
218         if(requestCode == SIGN_IN_GOOGLE){
219             handleSignInResult(Auth.GoogleSignInApi!!.
                    getSignInResultFromIntent(data))
220         }
221     }
222
223     private fun handleSignInResult(signInResultFromIntent:
            GoogleSignInResult?) {
224         if(signInResultFromIntent!!.isSuccess){
225             Toast.makeText(this, resources.getString(R.string.
                    str_sign_in_success), Toast.LENGTH_SHORT).show()
226             firebaseAuthWithGoogle(signInResultFromIntent.
                    signInAccount)
227         }
228         else{
229             Toast.makeText(this, resources.getString(R.string.
                    str_error_google_connection), Toast.LENGTH_SHORT).
                    show()
230         }
231     }
232
233
234     override fun onConnectionFailed(p0: ConnectionResult) {
235         Toast.makeText(this, resources.getString(R.string.
                str_error_google_connection), Toast.LENGTH_SHORT).show
                ()
236     }
```

```
237
238     private fun firebaseAuthWithGoogle(account:
            GoogleSignInAccount?){
239         val credential: AuthCredential = GoogleAuthProvider.
                getCredential(account!!.idToken, null)
240         mAuth!!.signInWithCredential(credential)
241                 .addOnCompleteListener{ taskSign ->
242                     if(taskSign.isSuccessful){
243                         //Success Login
244                         updateUI(mAuth!!.currentUser)
245                         Toast.makeText(this, resources.getString(
                                R.string.str_sign_in_success), Toast.
                                LENGTH_SHORT).show()
246                     }else{
247                         // If sign in fails, display a message to
                                the user.
248                         Toast.makeText(this, resources.getString(
                                R.string.str_error_google_connection),
                                Toast.LENGTH_SHORT).show()
249                     }
250                 }
251     }
252
253     ////////////////////  CHANGE ACTIVITY  ////////////////////
254     private fun updateUI(account: FirebaseUser?) {
255         DataStorePersistence.getInstance().setUser(
                applicationContext, account!!.uid, account.email)
256         checkPersistenceLists()
257         finish()
258     }
259
260     ////////////////////  LIFE CYCLE  ////////////////////
261     override fun onStart() {
262         super.onStart()
263         checkUserLogin()
264     }
265 }
```

Listing I.1: AuthenticationActivity.kt

I.1.2 Listas

```
1  public class MapsActivity extends BaseActivity {
2
3      ////////////////////  IMPLEMENT METHODS  ////////////////////
4      @Override
5      protected void initializeActionBar() {
6          actionbarLeftBtn.setVisibility(View.VISIBLE);
7      }
8
9      @Override
10     protected String getActionbarTitle() {
11         return getResources().getString(R.string.maps_activity);
12     }
13
14     @Override
15     protected int getContentView() {
16         return R.layout.maps_activity;
17     }
18
19     @Override
20     protected void assignViews() {
21
22     }
23
24     @Override
25     protected void prepareViews() {
26         MapsFragmentController.getInstance().attachToActivity(
               this, R.id.maps_container);
27     }
28
29     ////////////////////  BACK  ////////////////////
30     @Override
31     public void onBackPressed() {
32         if (!MapsFragmentController.getInstance().
               isFirstFragmentShown()) {
33             if(!MapsFragmentController.getInstance().isListChange
                   ()){
34                 DataStorePersistence.getInstance().
                       removeAddressListTemp(getApplicationContext());
```

```
35              }
36              MapsFragmentController.getInstance().setListChange(
                    false);
37              MapsFragmentController.getInstance().
                    showPreviousFragment();
38          } else {
39              super.onBackPressed();
40          }
41      }
42
43      ////////////////////  LIFE CYCLE  ////////////////////
44      @Override
45      protected void onDestroy() {
46          MapsFragmentController.getInstance().onDestroy();
47          super.onDestroy();
48      }
49
50      @Override
51      protected void onResume() {
52          super.onResume();
53          Crashlytics.log("MapsActivity on Resume");
54      }
55
56      @Override
57      protected void onPause() {
58          super.onPause();
59          Crashlytics.log("MapsActivity on Pause");
60      }
61  }
```

Listing I.2: MapsActivity

```
1   public class MapsMainFragment extends BaseFragment {
2
3       //In Layout
4       private CheckBox mapsMainMyLocationBtn;
5       private TextView mapsMainMyLocationText;
6       private SwipeRefreshLayout mapsMainRefresh;
7       private RecyclerView mapsMainList;
8       private Button mapsMainDistanceBtn;
9
```

```
10      //Not in Layout
11      private MapsMainListAdapter adapter;
12      private FirebaseAnalytics mFirebaseAnalytics;
13      private int selectedListsCount = 0;
14
15      //Location Elements
16      private FusedLocationProviderClient mFusedLocationClient;
17      private LocationRequest mLocationRequest;
18      private LocationCallback mLocationCallback;
19
20      private static final long UPDATE_INTERVAL_IN_MILLISECONDS =
            10000;
21      private static final long
            FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS =
22                UPDATE_INTERVAL_IN_MILLISECONDS / 2;
23
24      //Database
25      private FirebaseFirestore firestoreDatabase =
            FirebaseFirestore.getInstance();
26
27      //Activity Items
28      private ImageView actionbarAddIcon;
29      private ImageView actionbarIcon;
30      private TextView actionbarTitle;
31
32      ////////////////////  IMPLEMENT METHODS  ////////////////////
33      @Override
34      protected int getFragmentContentView() {
35          return R.layout.maps_main_fragment;
36      }
37
38      @Override
39      protected void assignViews() {
40          mapsMainMyLocationText = fragmentView.findViewById(R.id.
                maps_main_location_text);
41          mapsMainMyLocationBtn = fragmentView.findViewById(R.id.
                maps_main_location_checkbox);
42          mapsMainRefresh = fragmentView.findViewById(R.id.
                maps_main_swipe_refresh);
```

```
43          mapsMainList = fragmentView.findViewById(R.id.
               maps_main_list);
44          mapsMainDistanceBtn = fragmentView.findViewById(R.id.
               maps_main_distance_btn);
45
46          mFusedLocationClient = LocationServices.
               getFusedLocationProviderClient(Objects.requireNonNull(
               getActivity()));
47
48          // Obtain the Firebase Analytics instance.
49          if(getContext() != null) {
50              mFirebaseAnalytics = FirebaseAnalytics.getInstance(
                   getContext());
51          }
52          //activity actionbar
53          if (getActivity() != null) {
54              actionbarAddIcon = getActivity().findViewById(R.id.
                   actionbar_right_btn);
55              actionbarIcon = getActivity().findViewById(R.id.
                   actionbar_right_extra_btn);
56              actionbarTitle = getActivity().findViewById(R.id.
                   actionbar_text);
57          }
58      }
59
60      @Override
61      protected void prepareViews() {
62          selectedListsCount = 0;
63
64          //Location
65          createLocationRequest();
66          createLocationCallback();
67
68          mapsMainMyLocationText.setText(R.string.
               str_user_my_location);
69          mapsMainMyLocationBtn.setOnClickListener(new View.
               OnClickListener() {
70              @Override
71              public void onClick(View view) {
72                  if (Utils.isDoubleClick()) return;
```

```
73                  if (Utils.isGpsEnabled(Objects.requireNonNull(
                        getContext()))) {
74                      getLocation();
75                      if(mapsMainMyLocationBtn.isChecked()){
76                          selectList(true, DataStoreTemporary.
                                getInstance().getLocationAddress());
77                      }
78                      else{
79                          selectList(false, DataStoreTemporary.
                                getInstance().getLocationAddress());
80                      }
81                  }
82                  else{
83                      mapsMainMyLocationBtn.setChecked(false);
84                      selectList(false, DataStoreTemporary.
                            getInstance().getLocationAddress());
85                      locationSettingsDialog();
86                  }
87                  enableButton();
88              }
89          });
90
91          //Action Button
92          mapsMainDistanceBtn.setOnClickListener(new View.
                OnClickListener() {
93              @Override
94              public void onClick(View view) {
95                  if (Utils.isDoubleClick()) return;
96                  MapsFragmentController.getInstance().
                        showNextFragment(2);
97
98                  Bundle bundle = new Bundle();
99                  bundle.putString("Maps_Result", "test");
100                 mFirebaseAnalytics.logEvent("Button_result",
                        bundle);
101             }
102         });
103
104         //Refresh List
```

```
105         mapsMainRefresh.setColorSchemeResources(R.color.
               colorPrimary);
106         mapsMainRefresh.setOnRefreshListener(new
               SwipeRefreshLayout.OnRefreshListener() {
107             @Override
108             public void onRefresh() {
109                 getList(true);
110             }
111         });
112
113         //Start List
114         mapsMainList.setHasFixedSize(true);
115         int numberColumnsList = 2;
116         GridLayoutManager layoutManager = new GridLayoutManager(
               getContext(), numberColumnsList);
117         mapsMainList.setLayoutManager(layoutManager);
118         initAdapterIfNecessary();
119
120         //Add List
121         actionbarAddIcon.setOnClickListener(new View.
               OnClickListener() {
122             @Override
123             public void onClick(View view) {
124                 if (Utils.isDoubleClick()) return;
125                 if(DataStorePersistence.getInstance().
                       getUserEmail(getContext()).isEmpty()
126                         && !DataStorePersistence.getInstance().
                               getAddressListOne(getContext()).
                               getListID().isEmpty()
127                         && !DataStorePersistence.getInstance().
                               getAddressListTwo(getContext()).
                               getListID().isEmpty()){
128                     //User not logged and both list are not empty
129                     showErrorToast(getContext(), getResources().
                           getString(R.string.
                           str_maximum_guest_address_items));
130                 }
131                 else{
132                     //create new list
133                     addItemDialog(Properties.NEW_ID);
```

```
                }
            }
        });

        //Others
        enableButton();
        changeActionBar();
    }

    ////////////////////  FUNCTIONS  ////////////////////
    private void changeActionBar() {
        actionbarAddIcon.setVisibility(View.VISIBLE);
        actionbarAddIcon.setImageResource(R.drawable.
           ic_add_black_24dp);
        actionbarIcon.setVisibility(View.GONE);
        actionbarTitle.setText(R.string.maps_activity);
    }

    private void enableButton(){
        if(selectedListsCount == 2){
            mapsMainDistanceBtn.setEnabled(true);
        }
        else{
            mapsMainDistanceBtn.setEnabled(false);
        }
    }

    private void startAddressListFragment(String listID, String
       listTitle) {
        MapsFragmentController.getInstance().setListID(listID);
        MapsFragmentController.getInstance().setListTitle(
           listTitle);
        MapsFragmentController.getInstance().showNextFragment();
        if(getActivity()!= null) {
            getActivity().overridePendingTransition(R.animator.
               slide_in_left, R.animator.slide_out_right);
        }
    }

    ////////////////////  ITEMS_FUNCTIONS  ////////////////////
```

```
170
171     private void addItemDialog(final String listId){
172         final AlertDialog.Builder builder = new AlertDialog.
               Builder(Objects.requireNonNull(getContext()));
173         builder.setTitle(getString(R.string.str_list_name));
174
175         // Set up the input
176         final EditText input = new EditText(getContext());
177         // Specify the type of input expected; this, for example,
               sets the input as a password, and will mask the text
178         input.setInputType(InputType.TYPE_CLASS_TEXT);
179         builder.setView(input);
180
181         // Set up the buttons
182         builder.setPositiveButton(getString(R.string.str_next),
               new DialogInterface.OnClickListener() {
183             @Override
184             public void onClick(DialogInterface dialog, int which
                   ) {
185                 if (Utils.isDoubleClick()) return;
186                 if(input.length() > 0) {
187                     startAddressListFragment(listId, input.
                           getText().toString());
188                     dialog.dismiss();
189                 }
190                 else{
191                     showErrorToast(getContext(), getResources().
                           getString(R.string.str_enter_title));
192                 }
193                 showKeyboard(false, getContext());
194             }
195         });
196         builder.setNegativeButton(getString(R.string.str_discard)
               , new DialogInterface.OnClickListener() {
197             @Override
198             public void onClick(DialogInterface dialog, int which
                   ) {
199                 dialog.cancel();
200                 showKeyboard(false, getContext());
201             }
```

```
202         });
203
204         builder.show();
205         input.requestFocus();
206
207         showKeyboard(true, getContext());
208     }
209
210     private void checkItemList(int pos) {
211         if(selectedListsCount < 2 || adapter.getMapsMainItem(pos)
                .isCheck()) {
212             adapter.getMapsMainItem(pos).setCheck(!adapter.
                    getMapsMainItem(pos).isCheck());
213             if (adapter.getMapsMainItem(pos).isCheck()) {
214                 selectList(true, adapter.getMapsMainItem(pos).
                        getArrayAddressList());
215             } else {
216                 selectList(false, adapter.getMapsMainItem(pos).
                        getArrayAddressList());
217             }
218             enableButton();
219         }
220         else{
221             Toast.makeText(getContext(), getResources().getString
                    (R.string.str_only_two_lists), Toast.LENGTH_SHORT).
                    show();
222             adapter.getMapsMainItem(pos).setCheck(false);
223         }
224         adapter.updateListContent(adapter.
                getMapsMainItemArrayList());
225     }
226
227     private void openItemList(int pos) {
228         String listID = adapter.getMapsMainItem(pos).getListID();
229         String listTitle = adapter.getMapsMainItem(pos).
                getListTitle();
230
231         startAddressListFragment(listID, listTitle);
232
233         //Firebase Log
```

```
234         Bundle bundle = new Bundle();
235         bundle.putString("List_Option", listID);
236         mFirebaseAnalytics.logEvent("Button_List", bundle);
237     }
238 
239     ////////////////////  LIST FUNCTIONS  ////////////////////
240 
241     private void initAdapterIfNecessary() {
242         if (adapter == null) {
243             adapter = new MapsMainListAdapter();
244             adapter.setOnListItemClickedActionListener(new
                OnListItemClickedActionListener() {
245                 @Override
246                 public void onClicked(int pos, String action) {
247                     if (Utils.isDoubleClick()) return;
248                     switch (action) {
249                         case "openList":
250                             openItemList(pos);
251                             break;
252 
253                         case "selectItem":
254                             checkItemList(pos);
255                             break;
256 
257                         case "delete":
258                             deleteList(pos);
259                             break;
260                     }
261                 }
262             });
263         }
264         if (mapsMainList.getAdapter() == null) mapsMainList.
            setAdapter(adapter);
265     }
266 
267     private void getList(boolean reload){
268         if(reload) {
269             showLoadingDialog(true);
270             if (DataStorePersistence.getInstance().getUserEmail(
                getContext()).isEmpty()) {
```

```
271                 //User not logged
272                 ArrayList<JSONAddressList> mapsMainItemArrayList
                        = new ArrayList<>();
273
274                 if (!DataStorePersistence.getInstance().
                        getAddressListOne(getContext()).getListID().
                        isEmpty()) {
275                     //If exist list one
276                     mapsMainItemArrayList.add(new JSONAddressList
                            (DataStorePersistence.getInstance().
                            getAddressListOne(getContext()), false));
277                 }
278                 if (!DataStorePersistence.getInstance().
                        getAddressListTwo(getContext()).getListID().
                        isEmpty()) {
279                     //If exist list two
280                     mapsMainItemArrayList.add(new JSONAddressList
                            (DataStorePersistence.getInstance().
                            getAddressListTwo(getContext()), false));
281                 }
282
283                 updateList(mapsMainItemArrayList);
284             } else {
285                 //User logged
286
287                 firestoreDatabase.collection(Parameters.USER)
288                         .document(DataStorePersistence.
                                getInstance().getUserID(getContext()))
289                         .collection(Parameters.USER_LIST)
290                         .get()
291                         .addOnCompleteListener(new
                                OnCompleteListener<QuerySnapshot>() {
292                             @Override
293                             public void onComplete(@NonNull Task<
                                    QuerySnapshot> task) {
294                                 if (task.isSuccessful()) {
295                                     ArrayList<JSONAddressList>
                                            mapsMainItemArrayList = new
                                             ArrayList<>();
296
```

```
297                                     try {
298                                         for (DocumentSnapshot
                                               document : task.
                                               getResult().
                                               getDocuments()) {
299                                             mapsMainItemArrayList
                                                  .add(new
                                                  JSONAddressList(
                                                  Objects.
                                                  requireNonNull(
                                                  document.toObject(
                                                  JSONAddressList.
                                                  class)), false));
300                                         }
301                                     } catch (Exception e) {
302                                         showErrorToast(getContext
                                               (), e.getMessage());
303                                     }
304
305                                     updateList(
                                           mapsMainItemArrayList);
306                                 } else {
307                                     showErrorToast(getContext(),
                                           Objects.requireNonNull(task
                                           .getException()).getMessage
                                           ());
308                                     Crashlytics.logException(task
                                           .getException());
309                                 }
310                             }
311                         });
312             }
313             MapsFragmentController.getInstance().setReload(false)
                    ;
314         }
315     }
316
317     private void updateList(ArrayList<JSONAddressList>
            mapsMainItemArrayList){
318         //update
```

```
319         adapter.updateListContent(mapsMainItemArrayList);
320
321         showLoadingDialog(false);
322     }
323
324     private void selectList(boolean check, ArrayList<JSONAddress>
            jsonAddressArrayList){
325         if(check){
326             selectedListsCount++;
327             switch (selectedListsCount){
328                 case 1:
329                     MapsFragmentController.getInstance().
                            setJsonAddressArrayListOne(
                            jsonAddressArrayList);
330                     break;
331                 case 2:
332                     MapsFragmentController.getInstance().
                            setJsonAddressArrayListTwo(
                            jsonAddressArrayList);
333                     break;
334                 default:
335                     errorCleanSelect();
336                     break;
337             }
338         }
339         else {
340             if(selectedListsCount == 2 && !jsonAddressArrayList.
                    equals(
341                             MapsFragmentController.getInstance().
                                    getJsonAddressArrayListTwo())){
342                 //Change list two to list one.
343                 MapsFragmentController.getInstance().
                        setJsonAddressArrayListOne(
344                             MapsFragmentController.getInstance().
                                    getJsonAddressArrayListTwo());
345             }
346             selectedListsCount--;
347             if(selectedListsCount > 2 || selectedListsCount < 0 )
                    {
348                 errorCleanSelect();
```

```
            }
        }
    }

    private void deleteList(int pos) {
        switch (adapter.getMapsMainItem(pos).getListID()){
            case Properties.LIST_ONE_ID:
                DataStorePersistence.getInstance().
                   removeAddressListOne(getContext());
                break;
            case Properties.LIST_TWO_ID:
                DataStorePersistence.getInstance().
                   removeAddressListTwo(getContext());
                break;
            default:
                firestoreDatabase.collection(Parameters.USER)
                        .document(DataStorePersistence.
                           getInstance().getUserID(getContext()))
                        .collection(Parameters.USER_LIST)
                        .document(adapter.getMapsMainItem(pos).
                           getListID())
                        .delete()
                        .addOnSuccessListener(new
                           OnSuccessListener<Void>() {
                            @Override
                            public void onSuccess(Void aVoid) {
                                showSnackBar(R.string.delete);
                            }
                        })
                        .addOnFailureListener(new
                           OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull
                               Exception e) {
                                showSnackBar(R.string.
                                   error_try_again_later);
                            }
                        });
                break;
        }
```

```
381         selectList(false, adapter.getMapsMainItem(pos).
                getArrayAddressList());
382         adapter.getMapsMainItemArrayList().remove(pos);
383         adapter.updateListContent(adapter.
                getMapsMainItemArrayList());
384 
385     }
386 
387     private void errorCleanSelect() {
388         showLoadingDialog(true);
389         selectedListsCount = 0;
390         mapsMainMyLocationBtn.setChecked(false);
391         adapter.unselectAll();
392         showLoadingDialog(false);
393     }
394 
395     ////////////////////  LIFE CYCLE  ////////////////////
396     @Override
397     public void onStart() {
398         super.onStart();
399         Crashlytics.log("MapsMainFragment on Start");
400         getLocation();
401     }
402 
403     @Override
404     public void onResume() {
405         super.onResume();
406         getList(MapsFragmentController.getInstance().isReload());
407         errorCleanSelect();
408         Crashlytics.log("MapsMainFragment on Resume");
409     }
410     @Override
411     public void onPause() {
412         super.onPause();
413         Crashlytics.log("MapsMainFragment on Pause");
414 
415         //Remove Listener to avoid memory crash
416         mFusedLocationClient.removeLocationUpdates(
                mLocationCallback);
417     }
```

```
418
419     ////////////////////  LOCATION FUNCTIONS  ////////////////////
420
421     //Enable GPS in mobile
422     private void locationSettingsDialog() {
423         AlertDialog.Builder dialog = new AlertDialog.Builder(
                Objects.requireNonNull(getContext()));
424         dialog.setMessage(getResources().getString(R.string.
                str_gps_not_enable));
425         dialog.setPositiveButton(getResources().getString(R.
                string.str_open_location_settings), new DialogInterface
                .OnClickListener() {
426             @Override
427             public void onClick(DialogInterface
                    paramDialogInterface, int paramInt) {
428                 Intent myIntent = new Intent( Settings.
                        ACTION_LOCATION_SOURCE_SETTINGS);
429                 getContext().startActivity(myIntent);
430             }
431         });
432         dialog.setNegativeButton(getResources().getString(R.
                string.str_cancel), new DialogInterface.OnClickListener
                () {
433
434             @Override
435             public void onClick(DialogInterface
                    paramDialogInterface, int paramInt) {}
436         });
437         dialog.show();
438     }
439
440     private void getLocation() {
441         if (!checkPermissions()) {
442             requestPermissions();
443         } else {
444             getLastLocation();
445         }
446     }
447
448     @SuppressWarnings("MissingPermission")
```

```
449     private void getLastLocation() {
450         mFusedLocationClient.getLastLocation()
451                 .addOnCompleteListener(Objects.requireNonNull(
                        getActivity()), new OnCompleteListener<Location
                        >() {
452                     @Override
453                     public void onComplete(@NonNull Task<Location
                            > task) {
454                         if (task.isSuccessful() && task.getResult
                                () != null) {
455 
456                             setLocationUpdate(task.getResult());
457                         } else {
458                             Crashlytics.log("getLastLocation:
                                exception " +  task.getException())
                                ;
459                             showSnackBar(R.string.
                                str_no_location_detected);
460                             setLocationUncheck();
461                         }
462                     }
463                 });
464         mFusedLocationClient.requestLocationUpdates(
                mLocationRequest, mLocationCallback, Looper.myLooper())
                ;
465     }
466 
467     private void setLocationUncheck() {
468         mapsMainMyLocationBtn.setChecked(false);
469         selectList(false, DataStoreTemporary.getInstance().
                getLocationAddress());
470         showLoadingDialog(false);
471     }
472 
473     private void createLocationCallback() {
474         mLocationCallback = new LocationCallback() {
475             @Override
476             public void onLocationResult(LocationResult
                    locationResult) {
477                 super.onLocationResult(locationResult);
```

```
478
479                 setLocationUpdate(locationResult.getLastLocation
                        ());
480             }
481         };
482     }
483
484     private void setLocationUpdate(Location lastLocation) {
485         Properties.MY_CURRENT_LOCATION = lastLocation;
486         EventBus.getDefault().post(new EventLocationUpdate(
                Properties.MY_CURRENT_LOCATION));
487         setLocationText();
488     }
489
490     private void setLocationText() {
491         JSONLocation locationItem = Utils.getMyAddress(getContext
                (), Properties.MY_CURRENT_LOCATION.getLatitude(),
                Properties.MY_CURRENT_LOCATION.getLongitude());
492         if (locationItem == null) return;
493
494         ArrayList<JSONAddress> jsonAddressArrayList = new
                ArrayList<>();
495         jsonAddressArrayList.add(new JSONAddress(locationItem));
496         DataStoreTemporary.getInstance().setLocationAddress(
                jsonAddressArrayList);
497
498         String myLocation = locationItem.getAddress();
499
500         if(myLocation.equals("")){
501             mapsMainMyLocationText.setText(R.string.
                    str_user_my_location);
502         }
503         else {
504             mapsMainMyLocationText.setText(myLocation);
505         }
506     }
507
508     @SuppressLint("RestrictedApi")
509     private void createLocationRequest() {
510         mLocationRequest = new LocationRequest();
```

```
511
512         mLocationRequest.setInterval(
                UPDATE_INTERVAL_IN_MILLISECONDS);
513         mLocationRequest.setFastestInterval(
                FASTEST_UPDATE_INTERVAL_IN_MILLISECONDS);
514
515         mLocationRequest.setPriority(LocationRequest.
                PRIORITY_HIGH_ACCURACY);
516     }
517
518     ////////////////////  PERMISSIONS FUNCTIONS
            ////////////////////
519
520     private boolean checkPermissions() {
521         int permissionState = ActivityCompat.checkSelfPermission(
                Objects.requireNonNull(getActivity()),
522                 Manifest.permission.ACCESS_COARSE_LOCATION);
523         return permissionState == PackageManager.
                PERMISSION_GRANTED;
524     }
525
526     private void startLocationPermissionRequest() {
527         ActivityCompat.requestPermissions(Objects.requireNonNull(
                getActivity()),
528                 new String[]{Manifest.permission.
                    ACCESS_COARSE_LOCATION},
529                 Properties.RC_BASIC_LOCATION);
530     }
531
532     private void requestPermissions() {
533         boolean shouldProvideRationale =
534                 ActivityCompat.
                    shouldShowRequestPermissionRationale(Objects.
                    requireNonNull(getActivity()),
535                         Manifest.permission.
                            ACCESS_COARSE_LOCATION);
536
537         // Provide an additional rationale to the user. This
                would happen if the user denied the
```

```
538         // request previously, but didn't check the "Don't ask
                again" checkbox.
539         if (shouldProvideRationale) {
540             Crashlytics.log("Displaying permission rationale to
                    provide additional context.");
541 
542             showSnackBar(R.string.str_perm_location_rationale,
                    android.R.string.ok,
543                     new View.OnClickListener() {
544                         @Override
545                         public void onClick(View view) {
546                             // Request permission
547                             startLocationPermissionRequest();
548                         }
549                     });
550 
551         } else {
552             Crashlytics.log("Requesting permission");
553             startLocationPermissionRequest();
554         }
555     }
556 
557     @Override
558     public void onRequestPermissionsResult(int requestCode,
            @NonNull String[] permissions,
559                                            @NonNull int[]
                                                   grantResults) {
560 
561         Crashlytics.log("onRequestPermissionResult");
562         if (requestCode == Properties.RC_BASIC_LOCATION) {
563             if (grantResults.length <= 0) {
564                 // If user interaction was interrupted, the
                        permission request is cancelled and you
565                 // receive empty arrays.
566                 Crashlytics.log("User interaction was cancelled."
                        );
567             } else if (grantResults[0] == PackageManager.
                    PERMISSION_GRANTED) {
568                 // Permission granted.
569                 getLastLocation();
```

```
570             } else {
571                 // Permission denied.
572                 setLocationUncheck();
573 
574                 showSnackBar(R.string.
                        str_permission_denied_explanation, R.string.
                        str_permissions,
575                         new View.OnClickListener() {
576                             @Override
577                             public void onClick(View view) {
578                                 // Build intent that displays the
                                        App settings screen.
579                                 Intent intent = new Intent();
580                                 intent.setAction(
581                                         Settings.
                                            ACTION_APPLICATION_DETAILS_SETTINGS
                                            );
582                                 Uri uri = Uri.fromParts("package"
                                        ,
583                                         BuildConfig.
                                            APPLICATION_ID, null);
584                                 intent.setData(uri);
585                                 intent.setFlags(Intent.
                                        FLAG_ACTIVITY_NEW_TASK);
586                                 startActivity(intent);
587                             }
588                         });
589             }
590         }
591     }
592 }
```

Listing I.3: MapsMainFragment

```
1 public class MapsAddressListFragment extends BaseFragment {
2 
3     //In Layout
4     private SwipeRefreshLayout mapsListRefresh;
5     private RecyclerView mapsListRecycleView;
6     private TextView mapsListEmptyText;
7     private Button mapsListSaveBtn;
```

```
8
9       //Not in Layout
10      private MapsAddressListAdapter adapter;
11      private JSONAddressList addressListUndo = null;
12      private String listID;
13      private String listTitle;
14
15      //Activity Items
16      private FragmentActivity parentActivity = new
            FragmentActivity();
17      private ImageView actionbarSearchIcon;
18      private ImageView actionbarCleanIcon;
19      private TextView actionbarTitle;
20
21      //Google AutoComplete
22      private static final int PLACE_AUTOCOMPLETE_REQUEST_CODE = 1;
23      public static final int RESULT_OK           = -1;
24      public static final int RESULT_CANCELED     = 0;
25
26      //Database
27      FirebaseFirestore firebaseFirestore = FirebaseFirestore.
            getInstance();
28
29      ////////////////////  IMPLEMENT METHODS  ////////////////////
30      @Override
31      protected int getFragmentContentView() {
32          return R.layout.maps_address_list_fragment;
33      }
34
35
36      @Override
37      protected void assignViews() {
38          mapsListRefresh = fragmentView.findViewById(R.id.
                maps_list_swipe_refresh);
39          mapsListRecycleView = fragmentView.findViewById(R.id.
                maps_list_recycler_view);
40          mapsListEmptyText = fragmentView.findViewById(R.id.
                maps_list_empty_text);
41          mapsListSaveBtn = fragmentView.findViewById(R.id.
                maps_list_save_button);
```

```
42
43          listID = MapsFragmentController.getInstance().getListID()
                ;
44          listTitle = MapsFragmentController.getInstance().
                getListTitle();
45
46          if (getActivity() != null) {
47              parentActivity = getActivity();
48              actionbarSearchIcon = parentActivity.findViewById(R.
                    id.actionbar_right_btn);
49              actionbarCleanIcon = parentActivity.findViewById(R.id
                    .actionbar_right_extra_btn);
50              actionbarTitle = parentActivity.findViewById(R.id.
                    actionbar_text);
51          }
52      }
53
54      @Override
55      protected void prepareViews() {
56          actionbarSearchIcon.setOnClickListener(new View.
                OnClickListener() {
57              @Override
58              public void onClick(View v) {
59                  if (Utils.isDoubleClick()) return;
60                  try {
61                      AutocompleteFilter autocompleteFilter = new
                            AutocompleteFilter.Builder().build();
62
63                      Intent intent = new PlaceAutocomplete.
                            IntentBuilder(PlaceAutocomplete.
                            MODE_OVERLAY)
64                                  .setFilter(autocompleteFilter)
65                                  .build(parentActivity);
66
67                      startActivityForResult(intent,
                            PLACE_AUTOCOMPLETE_REQUEST_CODE);
68                  } catch (GooglePlayServicesRepairableException e)
                        {
```

```
69                    GoogleApiAvailability.getInstance().
                          getErrorDialog(getActivity(), e.
                          getConnectionStatusCode(),
70                              0 /* requestCode */).show();
71
72                    Crashlytics.logException(e);
73                } catch (GooglePlayServicesNotAvailableException
                      e) {
74                    String message = "Google Play Services is not
                          available" +
75                            GoogleApiAvailability.getInstance().
                                getErrorString(e.errorCode);
76                    Crashlytics.logException(e);
77                    Toast.makeText(getActivity(), message, Toast.
                          LENGTH_SHORT).show();
78                }
79            }
80        });
81
82        actionbarCleanIcon.setOnClickListener(new View.
              OnClickListener() {
83            @Override
84            public void onClick(View view) {
85                if (Utils.isDoubleClick()) return;
86                if(!DataStorePersistence.getInstance().
                      getAddressListTemp(getContext()).
                      getArrayAddressList().isEmpty()) {
87                    addressListUndo = DataStorePersistence.
                          getInstance().getAddressListTemp(getContext
                          ());
88                    DataStorePersistence.getInstance().
                          removeAddressListTemp(getContext());
89                    updateListContent();
90
91                    showSnackBar(R.string.str_list_cleaned, R.
                          string.str_undo,
92                            new View.OnClickListener() {
93                                @Override
94                                public void onClick(View view) {
95                                    // Request permission
```

```
96                                      DataStorePersistence.
                                           getInstance().
                                           setAddressListTemp(
                                           getContext(),
                                           addressListUndo);
97                                      updateListContent();
98                                      showSnackBar(R.string.
                                           str_list_restored);
99                                  }
100                             });
101             }
102             else{
103                 Toast.makeText(getContext(), getResources().
                        getString(R.string.str_no_list_to_clean),
                        Toast.LENGTH_SHORT).show();
104             }
105         }
106     });
107
108     mapsListSaveBtn.setOnClickListener(new View.
            OnClickListener() {
109         @Override
110         public void onClick(View view) {
111             if(Utils.isDoubleClick()) return;
112             if(checkNotEmptyList()){
113                 if(DataStorePersistence.getInstance().
                        getUserEmail(getContext()).isEmpty()) {
114                     //not logged
115                     saveInMobile();
116                 }
117                 else{
118                     //logged
119                     saveInDatabase();
120                 }
121                 //Clean Temp list
122                 DataStorePersistence.getInstance().
                        removeAddressListTemp(getContext());
123                 MapsFragmentController.getInstance().
                        showPreviousFragment();
```

```
124                     MapsFragmentController.getInstance().
                           setListChange(false);
125                     MapsFragmentController.getInstance().
                           setReload(true);
126                 }
127             }
128         });
129 
130         mapsListRefresh.setColorSchemeResources(R.color.
               colorPrimary);
131         mapsListRefresh.setOnRefreshListener(new
               SwipeRefreshLayout.OnRefreshListener() {
132             @Override
133             public void onRefresh() {
134                 updateListContent();
135             }
136         });
137 
138         //List
139         mapsListRecycleView.setHasFixedSize(true);
140         LinearLayoutManager linearLayoutManager = new
               LinearLayoutManager(getContext());
141         mapsListRecycleView.setLayoutManager(linearLayoutManager)
               ;
142         initAdapterIfNecessary();
143 
144         //Actionbar
145         changeActionBar();
146     }
147 
148     ////////////////////  SAVE FUNCTIONS  ////////////////////
149     private boolean checkNotEmptyList() {
150         if(DataStorePersistence.getInstance().getAddressListTemp(
               getContext()).getArrayAddressList().isEmpty()){
151             showErrorToast(getContext(), R.string.
                   error_empty_list);
152             return false;
153         }
154         return true;
155     }
```

```
156
157     private void saveInMobile() {
158         //if new and list one is empty
159         if(listID.equals(Properties.NEW_ID) &&
               DataStorePersistence.getInstance().getAddressListOne(
               getContext()).getListID().isEmpty()){
160             listID = Properties.LIST_ONE_ID;
161         }
162         //if new and list two is empty
163         else if(listID.equals(Properties.NEW_ID) &&
               DataStorePersistence.getInstance().getAddressListTwo(
               getContext()).getListID().isEmpty()){
164             listID = Properties.LIST_TWO_ID;
165         }
166         DataStorePersistence.getInstance().getAddressListTemp(
               getContext()).setListID(listID);
167         //save
168         switch (listID) {
169             case Properties.NEW_ID:
170                     showErrorToast(getContext(), getResources().
                           getString(R.string.
                           str_maximum_guest_address_items));
171                 break;
172             case Properties.LIST_ONE_ID:
173                 DataStorePersistence.getInstance()
174                         .setAddressListOne(getContext(),
175                                 DataStorePersistence.getInstance
                                       ()
176                                         .getAddressListTemp(
                                           getContext()));
177                 break;
178             case Properties.LIST_TWO_ID:
179                 DataStorePersistence.getInstance()
180                         .setAddressListTwo(getContext(),
181                                 DataStorePersistence.getInstance
                                       ()
182                                         .getAddressListTemp(
                                           getContext()));
183                 break;
184         }
```

```
185     }
186
187     private void saveInDatabase() {
188
189         //Check if Address List is new
190         if(listID.equals(Properties.NEW_ID)){
191
192             //Create a new document on DB
193             DocumentReference newCityRef = firebaseFirestore.
                    collection(Parameters.USER)
194                     .document(DataStorePersistence.getInstance().
                            getUserID(getContext()))
195                     .collection(Parameters.USER_LIST)
196                     .document();
197
198             //Get new document ID
199             listID = newCityRef.getId();
200             DataStorePersistence.getInstance().getAddressListTemp
                    (getContext()).setListID(listID);
201         }
202
203         //Save Address List on DB
204         firebaseFirestore.collection(Parameters.USER)
205                 .document(DataStorePersistence.getInstance().
                        getUserID(getContext()))
206                 .collection(Parameters.USER_LIST)
207                 .document(listID)
208                 .set(DataStorePersistence.getInstance().
                        getAddressListTemp(getContext()))
209                 .addOnCompleteListener(new OnCompleteListener<
                        Void>() {
210                     @Override
211                     public void onComplete(@NonNull Task<Void>
                            task) {
212                         if(!task.isSuccessful()){
213                             showErrorToast(getContext(), Objects.
                                    requireNonNull(task.getException())
                                    .getMessage());
214                             Crashlytics.logException(task.
                                    getException());
```

```
                        }
                    }
                });
    }

    ////////////////////  ADAPTER LIST  ////////////////////
    private void getList() {
        //check if there is a temp list saved
        if(!DataStorePersistence.getInstance().getAddressListTemp
           (getContext()).getListID().isEmpty()){
            showDialog(DataStorePersistence.getInstance().
               getAddressListTemp(getContext()).getListID().equals
               (listID));
        }
        else{
            showLoadingDialog(true);
            populateList();
        }
    }

    private void populateList() {
        switch (listID){
            case Properties.NEW_ID:
                DataStorePersistence.getInstance().
                   setAddressListTemp(getContext(),
                        new JSONAddressList());
                updateListContent();
                break;
            case Properties.LIST_ONE_ID:
                DataStorePersistence.getInstance().
                   setAddressListTemp(getContext(),
                        DataStorePersistence.getInstance().
                           getAddressListOne(getContext())
                );
                updateListContent();
                break;

            case Properties.LIST_TWO_ID:
                DataStorePersistence.getInstance().
                   setAddressListTemp(getContext(),
```

```
248                         DataStorePersistence.getInstance().
                               getAddressListTwo(getContext())
249                 );
250                 updateListContent();
251                 break;
252             default:
253                 firebaseFirestore.collection(Parameters.USER)
254                         .document(DataStorePersistence.
                               getInstance().getUserID(getContext()))
255                         .collection(Parameters.USER_LIST)
256                         .document(listID)
257                         .get()
258                         .addOnCompleteListener(new
                               OnCompleteListener<DocumentSnapshot>()
                               {
259                             @Override
260                             public void onComplete(@NonNull Task<
                                   DocumentSnapshot> task) {
261                                 if (task.isSuccessful()) {
262                                     try {
263                                         DataStorePersistence.
                                               getInstance().
                                               setAddressListTemp(
                                               getContext(), task.
                                               getResult().toObject(
                                               JSONAddressList.class))
                                               ;
264                                         updateListContent();
265                                     } catch (Exception e) {
266                                         showErrorToast(getContext
                                               (), e.getMessage());
267                                     }
268                                 } else {
269                                     showErrorToast(getContext(),
                                           Objects.requireNonNull(task
                                           .getException()).getMessage
                                           ());
270                                     Crashlytics.logException(task
                                           .getException());
271                                 }
```

```
272                             }
273                         });
274                 break;
275         }
276         DataStorePersistence.getInstance().getAddressListTemp(
               getContext()).setListID(listID);
277         DataStorePersistence.getInstance().getAddressListTemp(
               getContext()).setListTitle(listTitle);
278     }
279
280     private void initAdapterIfNecessary() {
281         if (adapter == null) {
282             adapter = new MapsAddressListAdapter();
283             adapter.setOnListItemClickedActionListener(new
                   OnListItemClickedActionListener() {
284                 @Override
285                 public void onClicked(int pos, String action) {
286                     if (Utils.isDoubleClick()) return;
287                     switch (action) {
288                         case "remove":
289                             removeAddressFromList(pos);
290                             break;
291                     }
292
293                 }
294             });
295         }
296         if (mapsListRecycleView.getAdapter() == null)
297             mapsListRecycleView.setAdapter(adapter);
298     }
299
300     public void updateListContent() {
301         ArrayList<JSONAddress> jsonAddresses;
302
303         jsonAddresses = DataStorePersistence.getInstance().
               getAddressListTemp(getContext()).getArrayAddressList();
304
305         initAdapterIfNecessary();
306         adapter.updateListContent(jsonAddresses);
307
```

```
308         if (adapter.getItemCount() == 0) {
309             mapsListEmptyText.setVisibility(View.VISIBLE);
310         } else {
311             mapsListEmptyText.setVisibility(View.GONE);
312         }
313         stopRefresh();
314         showLoadingDialog(false);
315     }
316 
317     private void stopRefresh() {
318         if (mapsListRefresh != null && mapsListRefresh.
               isRefreshing())
319             mapsListRefresh.setRefreshing(false);
320     }
321 
322     private void removeAddressFromList(int pos) {
323         DataStorePersistence.getInstance().getAddressListTemp(
               getContext()).getArrayAddressList().remove(pos);
324 
325         updateListContent();
326         MapsFragmentController.getInstance().setListChange(true);
327     }
328 
329     ////////////////////  DIALOG  ////////////////////
330     protected void showDialog(final boolean sameList) {
331         //Open Suggestion dialog
332         View dialogView =  LayoutInflater.from(getContext()).
               inflate(R.layout.view_dialog_text_two_option_layout,
               null, false);
333 
334         TextView dialogTitle = dialogView.findViewById(R.id.
               dialog_title_text);
335         TextView dialogDescription = dialogView.findViewById(R.id
               .dialog_description_text);
336         Button dialogCancelBtn = dialogView.findViewById(R.id.
               dialog_cancel_btn);
337         Button dialogOkBtn = dialogView.findViewById(R.id.
               dialog_ok_btn);
338 
339         if(sameList){
```

```
340             dialogDescription.setText(R.string.str_keep_editing);
341             dialogOkBtn.setText(R.string.str_keep);
342         }
343         else{
344             String description = getString(R.string.
                    str_keep_edition_other_list,
345                         DataStorePersistence.getInstance().
                                getAddressListTemp(getContext()).
                                getListTitle());
346             dialogDescription.setText(description);
347             dialogOkBtn.setText(R.string.str_go_back);
348 
349         }
350         dialogTitle.setText(R.string.str_unsaved_edit);
351         dialogCancelBtn.setText(R.string.str_discard);
352 
353         final AlertDialog dialog = new AlertDialog.Builder(
                Objects.requireNonNull(getContext())).create();
354         dialog.setView(dialogView);
355         dialog.setCancelable(false);
356 
357         dialogCancelBtn.setOnClickListener(new View.
                OnClickListener() {
358             @Override
359             public void onClick(View v) {
360                 if (Utils.isDoubleClick()) return;
361                 dialog.dismiss();
362                 populateList();
363             }
364         });
365 
366         dialogOkBtn.setOnClickListener(new View.OnClickListener()
                {
367             @Override
368             public void onClick(View v) {
369                 if (Utils.isDoubleClick()) return;
370                 dialog.dismiss();
371                 updateListContent();
372                 listID = DataStorePersistence.getInstance().
                            getAddressListTemp(getContext()).getListID();
```

```
373                 listTitle = DataStorePersistence.getInstance().
                        getAddressListTemp(getContext()).getListTitle()
                        ;
374                 changeActionBar();
375                 MapsFragmentController.getInstance().
                        setListChange(true);
376             }
377         });
378 
379 
380         dialog.show();
381     }
382 
383     ////////////////////  ACTION BAR  ////////////////////
384     private void changeActionBar() {
385         actionbarSearchIcon.setVisibility(View.VISIBLE);
386         actionbarSearchIcon.setImageResource(R.drawable.
                ic_search_black_24dp);
387 
388         actionbarCleanIcon.setVisibility(View.VISIBLE);
389         actionbarCleanIcon.setImageResource(R.drawable.
                ic_clear_all_black_24dp);
390 
391         actionbarTitle.setText(listTitle);
392     }
393 
394     ////////////////////  LIFE CYCLE  ////////////////////
395     @Override
396     public void onStart() {
397         super.onStart();
398         getList();
399         Crashlytics.log("MapsAddressListFragment on Start");
400     }
401 
402     @Override
403     public void onResume() {
404         super.onResume();
405         Crashlytics.log("MapsAddressListFragment on Resume");
406     }
407     @Override
```

```
408     public void onPause() {
409         super.onPause();
410         Crashlytics.log("MapsAddressListFragment on Pause");
411     }
412 
413     ////////////////////  GOOGLE API RESULT  ////////////////////
414     @Override
415     public void onActivityResult(int requestCode, int resultCode,
            Intent data) {
416         if (requestCode == PLACE_AUTOCOMPLETE_REQUEST_CODE) {
417             if (resultCode == RESULT_OK) {
418                 JSONAddress newAddress = new JSONAddress(
                        PlaceAutocomplete.getPlace(parentActivity.
                        getApplicationContext(), data));
419 
420                 DataStorePersistence.getInstance().
                        getAddressListTemp(getContext()).
                        getArrayAddressList().add(newAddress);
421 
422                 updateListContent();
423                 MapsFragmentController.getInstance().
                        setListChange(true);
424 
425             } else if (resultCode == PlaceAutocomplete.
                    RESULT_ERROR) {
426                 Status status = PlaceAutocomplete.getStatus(
                        parentActivity.getApplicationContext(), data);
427                 showErrorToast(getContext(), status.
                        getStatusMessage());
428 
429             } else if (resultCode == RESULT_CANCELED) {
430                 Crashlytics.log("Please Autocomplete Canceled");
431             }
432         }
433 
434     }
435 
436 }
```

Listing I.4: MapsAddressListFragment

```
public class MapsResultListFragment extends BaseFragment {

    //In Layout
    private SwipeRefreshLayout mapsDistanceListRefresh;
    private RecyclerView mapsDistanceList;

    //Not in Layout
    private MapsResultListAdapter adapter;
    private LinearLayoutManager linearLayoutManager;
    private ArrayList<JSONAddressDistance>
       jsonAddressDistanceArrayListHolder = new ArrayList<>();

    //Activity Items
    private FragmentActivity parentActivity = new
       FragmentActivity();
    private TextView actionbarTitle;
    private ImageView actionbarFilterIcon;

    //List Filters
    private boolean hideNoResponse;

    ////////////////////  IMPLEMENT METHODS  ////////////////////
    @Override
    protected int getFragmentContentView() { return R.layout.
       maps_result_list_fragment; }

    @Override
    protected void assignViews() {
        mapsDistanceListRefresh = fragmentView.findViewById(R.id.
           maps_result_swipe_refresh);
        mapsDistanceList = fragmentView.findViewById(R.id.
           maps_result_recycler_view);

        if (getActivity() != null) {
            parentActivity = getActivity();
            actionbarTitle = parentActivity.findViewById(R.id.
               actionbar_text);
            actionbarFilterIcon = parentActivity.findViewById(R.
               id.actionbar_right_btn);
        }
```

```
34      }
35
36      @Override
37      protected void prepareViews() {
38          mapsDistanceListRefresh.setColorSchemeResources(R.color.
               colorPrimary);
39          mapsDistanceListRefresh.setOnRefreshListener(new
               SwipeRefreshLayout.OnRefreshListener() {
40              @Override
41              public void onRefresh() {
42                  loadData(true);
43              }
44          });
45
46          //List
47          mapsDistanceList.setHasFixedSize(true);
48          linearLayoutManager = new LinearLayoutManager(getContext
               ());
49          mapsDistanceList.setLayoutManager(linearLayoutManager);
50          initAdapterIfNecessary();
51
52          mapsDistanceList.addOnScrollListener(new RecyclerView.
               OnScrollListener() {
53              boolean isSlidingToLast = false;
54              int lastVisibleItemPosition = 0;
55              int lastPositions = 0;
56
57              @Override
58              public void onScrollStateChanged(RecyclerView
                   recyclerView, int newState) {
59                  if (newState == RecyclerView.SCROLL_STATE_IDLE &&
                       linearLayoutManager.getItemCount() > 0) {
60                      lastPositions = linearLayoutManager.
                           getItemCount();
61                      lastVisibleItemPosition = linearLayoutManager
                           .findLastVisibleItemPosition();
62
63                      if (lastPositions - 1 ==
                           lastVisibleItemPosition && isSlidingToLast)
                           {
```

```
                        loadData(false);
                    }
                }
            }

            @Override
            public void onScrolled(RecyclerView recyclerView, int
                dx, int dy) {
                super.onScrolled(recyclerView, dx, dy);
                isSlidingToLast = dy > 0;
            }
        });

        actionbarFilterIcon.setOnClickListener(new View.
           OnClickListener() {
            @Override
            public void onClick(View view) {
                if(Utils.isDoubleClick()) return;
                openFilterDialog();
            }
        });

        //Actionbar
        changeActionBar();
    }

    ////////////////////  ACTION BAR  ////////////////////
    private void changeActionBar() {
        actionbarTitle.setText(R.string.maps_activity);
        actionbarFilterIcon.setVisibility(View.VISIBLE);
        actionbarFilterIcon.setImageResource(R.drawable.
           ic_filter_list_black_24dp);
    }

    ////////////////////  ADAPTER LIST  ////////////////////
    private void initAdapterIfNecessary() {
        if (adapter == null) {
            adapter = new MapsResultListAdapter();
            adapter.setOnListItemClickedActionListener(new
               OnListItemClickedActionListener() {
```

```
100                 @Override
101                 public void onClicked(int pos, String action) {
102                     if (Utils.isDoubleClick()) return;
103                     switch (action) {
104                         case "open":
105                             openMaps(pos);
106                             break;
107                     }
108 
109                 }
110             });
111         }
112         if (mapsDistanceList.getAdapter() == null)
113             mapsDistanceList.setAdapter(adapter);
114     }
115 
116     public void updateListContent(ArrayList<JSONAddressDistance>
        jsonAddressDistanceArrayList) {
117         initAdapterIfNecessary();
118         adapter.updateListContent(jsonAddressDistanceArrayList);
119         showLoadingDialog(false);
120     }
121 
122     private void stopRefresh() {
123         if (mapsDistanceListRefresh != null &&
            mapsDistanceListRefresh.isRefreshing())
124             mapsDistanceListRefresh.setRefreshing(false);
125     }
126 
127     ////////////////////  START_ACTIVITY  ////////////////////
128 
129     private void openMaps(int pos){
130         String origin, destination, url;
131         try {
132             origin = URLEncoder.encode(adapter.getItem(pos).
                getLocationOne(), "utf-8");
133             destination = URLEncoder.encode(adapter.getItem(pos).
                getLocationTwo(), "utf-8");
```

```
134             url = Properties.googleMapsDirectionURL + "&origin="
                    + origin + "&destination=" + destination + "&
                    travelmode=" + Parameters.TRAVEL_MODE;
135
136             startActivity(new Intent(android.content.Intent.
                    ACTION_VIEW, Uri.parse(url)));
137
138             Objects.requireNonNull(getActivity()).
                    overridePendingTransition(R.animator.slide_in_right
                    , R.animator.slide_out_left);
139
140         } catch (UnsupportedEncodingException e) {
141             e.printStackTrace();
142         }
143     }
144
145     ////////////////////  LIST_FILTER  ////////////////////
146     private void openFilterDialog() {
147         final AlertDialog.Builder builder = new AlertDialog.
                Builder(Objects.requireNonNull(getContext()));
148         builder.setTitle(getString(R.string.filter));
149
150         // Set up the input
151         final CheckBox hideNoResponseBox = new CheckBox(
                getContext());
152         //Put the checkBox on the right side
153         hideNoResponseBox.setLayoutDirection(View.
                LAYOUT_DIRECTION_RTL);
154         hideNoResponseBox.setText(R.string.hide_no_responses);
155         hideNoResponseBox.setChecked(hideNoResponse);
156         hideNoResponseBox.setWidth(WindowManager.LayoutParams.
                WRAP_CONTENT);
157
158         hideNoResponseBox.setGravity(Gravity.START | Gravity.
                CENTER_VERTICAL);
159         hideNoResponseBox.setTextAlignment(View.
                TEXT_ALIGNMENT_TEXT_START);
160
161         int px = Math.round(TypedValue.applyDimension(
```

```
                TypedValue.COMPLEX_UNIT_DIP, 16, getResources().
                    getDisplayMetrics()));

        hideNoResponseBox.setPadding(px,0,0,0);

        builder.setView(hideNoResponseBox);


        // Set up the buttons
        builder.setPositiveButton(getString(R.string.do_filter),
            new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which
                ) {
                if (Utils.isDoubleClick()) return;
                hideNoResponse = hideNoResponseBox.isChecked();

                showLoadingDialog(true);
                hideListItem(adapter.getDistanceArrayList());
                dialog.dismiss();
            }
        });
        builder.setNegativeButton(getString(R.string.str_cancel),
            new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which
                ) {
                dialog.cancel();
            }
        });

        builder.show();
    }

    private void hideListItem(ArrayList<JSONAddressDistance>
        addressDistanceArrayListComplete) {
        addressDistanceArrayListComplete.addAll(
            jsonAddressDistanceArrayListHolder);

        jsonAddressDistanceArrayListHolder = new ArrayList<>();
```

```
195
196         for(JSONAddressDistance addressDistance:
                addressDistanceArrayListComplete) {
197             boolean saved = false;
198             if(hideNoResponse){ //Hide No Response elements
199                 if(!addressDistance.isSuccess() && !saved){ //Is
                        fail response (no response) and is not remove
                        yet
200                     jsonAddressDistanceArrayListHolder.add(
                            addressDistance);
201                     saved = true;
202                 }
203             }
204             //Todo if city
205         }
206         //Remove Hide Elements
207         addressDistanceArrayListComplete.removeAll(
                jsonAddressDistanceArrayListHolder);
208
209         //Update List
210         updateListContent(addressDistanceArrayListComplete);
211     }
212
213     ////////////////////  LIFE CYCLE  ////////////////////
214     @Override
215     public void onStart() {
216         super.onStart();
217         if (!EventBus.getDefault().isRegistered(this)) EventBus.
                getDefault().register(this);
218     }
219
220     @Override
221     public void onResume() {
222         super.onResume();
223         loadData(true);
224         Crashlytics.log("MapsResultListFragment on Resume");
225     }
226
227     @Override
228     public void onPause() {
```

```
229         super.onPause();
230         Crashlytics.log("MapsResultListFragment on Pause");
231     }
232 
233     @Override
234     public void onStop() {
235         if (EventBus.getDefault().isRegistered(this)) EventBus.
               getDefault().unregister(this);
236         super.onStop();
237     }
238 
239     ////////////////////  DATA  ////////////////////
240     private void loadData(boolean reload) {
241         if(reload) showLoadingDialog(true);
242 
243         new ApiCallGetAddressDistance(parentActivity
244                 .getApplicationContext())
245                 .getDistance(reload,
246                         MapsFragmentController.getInstance().
                            getJsonAddressArrayListOne(),
247                         MapsFragmentController.getInstance().
                            getJsonAddressArrayListTwo());
248     }
249 
250     ////////////////////  EVENT  ////////////////////
251     @Subscribe(threadMode = ThreadMode.MAIN)
252     public void onMessageEvent(EventGetAddressDistances event) {
253         stopRefresh();
254         showLoadingDialog(false);
255         if(event.code.equals(ResponseProperties.SUCCESS)) {
256             jsonAddressDistanceArrayListHolder = new ArrayList
                   <>();
257             hideListItem(event.addressDistances);
258             loadData(false);
259         }
260         if(event.message.length() > 2){
261             showErrorToast(getContext(), event.message);
262         }
263     }
264 
```

```
265 }
```

Listing I.5: MapsResultListFragment

I.2 Chamada de API

```
1  public class MessageReceivedService extends
       FirebaseMessagingService {
2      public MessageReceivedService() {
3      }
4
5      @Override
6      public void onMessageReceived(RemoteMessage remoteMessage) {
7          // Not getting messages here? See why this may be: https
               ://goo.gl/39bRNJ
8          FirebaseCrash.log( "From: " + remoteMessage.getFrom());
9
10         // Check if message contains a data payload.
11         if (remoteMessage.getData().size() > 0) {
12             FirebaseCrash.log( "Message data payload: " +
                   remoteMessage.getData());
13         }
14
15
16         // Check if message contains a notification payload.
17         if (remoteMessage.getNotification() != null) {
18             FirebaseCrash.log( "Message Notification Body: " +
                   remoteMessage.getNotification().getBody());
19         }
20
21         new Intent(this, MainActivity.class);
22     }
23 }
```

Listing I.6: MessageReceivedService

```
1  public class TokenService extends FirebaseInstanceIdService {
2
3      @Override
4      public void onTokenRefresh() {
5          // Get updated InstanceID token.
```

```
        String refreshedToken = FirebaseInstanceId.getInstance().
            getToken();
        FirebaseCrash.log( "Refreshed token: " + refreshedToken);


        // If you want to send messages to this application
            instance or
        // manage this apps subscriptions on the server side,
            send the
        // Instance ID token to your app server.
        Properties.CURRENT_TOKEN = refreshedToken;
        //TODO sendRegistrationToServer(refreshedToken);
    }
}
```

Listing I.7: TokenService

```
public interface ApiCallGoogle {
    @GET("{path}")
    Call<ResponseBody> get(
        @Path(value = "path", encoded = true) String path,
        @QueryMap(encoded = true) Map<String, String> params); //
            encoded = true make " " became %20""

    @POST("TODO")
    Call<ResponseBody> post(
        @QueryMap Map<String, String> params);


    Retrofit retrofit = new Retrofit.Builder()
            .baseUrl(Properties.googleMapsApiURL)
            .addConverterFactory(GsonConverterFactory.create())
            .build();
}
```

Listing I.8: ApiCallGoogle

```
public abstract class ApiCallBase {
    protected String action = "";
    protected String key = Utils.generateID();
    protected Context context;

```

```
6       ////////////////////////////////// ABSTRACT METHODS
            ////////////////////////////////////
7       protected abstract void sendEvent(String code, String message
            ); //Send a Event after Api Call
8       protected abstract void apiResponse(JSONObject object); //
            Treat the response from the call
9
10      ////////////////////////////////// Connection
            ////////////////////////////////////
11      protected void startGoogleGetConnection(String action, Map<
            String, String> params, final String key) {
12          ApiCallGoogle apiCallGoogle = ApiCallGoogle.retrofit.
                create(ApiCallGoogle.class);
13
14          final Call<ResponseBody> call = apiCallGoogle.get(action,
                params);
15
16          if(!isRunning(key)) {
17              try{
18                  keyRunning(key);
19                  call.enqueue(new Callback<ResponseBody>() {
20                      @Override
21                      public void onResponse(@NotNull Call<
                            ResponseBody> call, @NotNull Response<
                            ResponseBody> response) {
22                          keyNotRunning(key);
23                          String detailsString =
                                getStringFromRetrofitResponse(response)
                                ;
24                          JSONObject jsonResponse;
25                          try {
26                              jsonResponse = new JSONObject(
                                    detailsString);
27                              apiResponse(jsonResponse);
28                          } catch (JSONException e) {
29                              e.printStackTrace();
30                              FirebaseCrash.report(e);
31                          }
32                      }
33                      @Override
```

```
34                    public void onFailure(@NotNull Call<
                          ResponseBody> call, @NotNull Throwable t) {
35                        keyNotRunning(key);
36                        sendEvent(ResponseProperties.CALL_FAIL,
                              ResponseProperties.getMessageError(
                              context, ResponseProperties.CALL_FAIL))
                              ;
37                    }
38                });
39            }
40            catch (Exception e) {
41                FirebaseCrash.report(e);
42                keyNotRunning(key);
43                e.printStackTrace();
44                sendEvent(ResponseProperties.CONNECTION_FAIL,
                      ResponseProperties.getMessageError(context,
                      ResponseProperties.CONNECTION_FAIL));
45            }
46        }
47    }
48
49    ////////////////////////////////// TREAT RESPONSE
          ////////////////////////////////
50    private static String getStringFromRetrofitResponse(@NotNull
          Response<ResponseBody> response) {
51        //Try to get response body
52        BufferedReader reader;
53        StringBuilder sb = new StringBuilder();
54        ResponseBody responseBody = response.body();
55        if(responseBody != null) {
56            try (InputStream inputStream = responseBody.
                  byteStream()) {
57                reader = new BufferedReader(new InputStreamReader
                      (inputStream));
58                String line;
59
60                try {
61                    while ((line = reader.readLine()) != null) {
62                        sb.append(line);
63                    }
```

```
64                  } catch (IOException e) {
65                      e.printStackTrace();
66                      FirebaseCrash.report(e);
67                  }
68              } catch (IOException e) {
69                  e.printStackTrace();
70                  FirebaseCrash.report(e);
71              }
72          }
73
74          return sb.toString();
75
76      }
77
78      ////////////////////////////////// RUNNING KEYS
           ////////////////////////////////////
79      //This keys avoid the same API Call been used again before it
           get a answer (like the user click 3 times)
80
81      private void keyNotRunning(String key) {
82          if (DataStoreTemporary.getInstance().getRunningKeys() !=
               null && DataStoreTemporary.getInstance().getRunningKeys
               ().containsKey(key)) {
83              DataStoreTemporary.getInstance().getRunningKeys().put
                   (key, false);
84          }
85      }
86
87      private void keyRunning(String key) {
88          if (DataStoreTemporary.getInstance().getRunningKeys() !=
               null) {
89              DataStoreTemporary.getInstance().getRunningKeys().put
                   (key, true);
90          }
91      }
92
93      private boolean isRunning(String key) {
94          return DataStoreTemporary.getInstance().getRunningKeys()
               != null &&
```

```
95              !DataStoreTemporary.getInstance().getRunningKeys().
                    isEmpty() &&
96              DataStoreTemporary.getInstance().getRunningKeys().
                    containsKey(key);
97      }
98  }
```

Listing I.9: ApiCallBase

```
1
2   public class ApiCallGetAddressDistance extends ApiCallBase {
3       private ArrayList<JSONAddress> addressesListOne = new
            ArrayList<>();
4       private ArrayList<JSONAddress> addressesListTwo = new
            ArrayList<>();
5       private ArrayList<JSONAddressDistance>
            arrayJsonAddressDistance = null;
6       private JSONPaginatorList jsonPaginatorList = null;
7       private StringBuilder listA;
8       private StringBuilder listB;
9       private boolean reload = false;
10
11      public ApiCallGetAddressDistance (Context context){
12          this.context = context;
13          action = Paths.googleDistanceMatrix;
14
15      }
16
17      ////////////////  Setup params  ////////////////
18      public void getDistance(boolean reload, ArrayList<JSONAddress
            > listOne, ArrayList<JSONAddress> listTwo){
19          this.reload = reload;
20          addressesListOne = listOne;
21          addressesListTwo = listTwo;
22
23          if(setupPaginator()) {
24              getLists();
25
26              Map<String, String> params = new HashMap<>();
27              params.put(Parameters.ORIGIN, listA.toString());
28              params.put(Parameters.DESTINATION, listB.toString());
```

```
29              params.put(Parameters.MODE, Parameters.TRAVEL_MODE);
30              params.put(Parameters.KEY, Parameters.API_KEY);
31
32              startGoogleGetConnection(action, params, key);
33          }
34      }
35
36      ////////////////  Functions  ////////////////
37      private boolean setupPaginator() {
38          if(jsonPaginatorList == null){
39              jsonPaginatorList = new JSONPaginatorList();
40          }
41          if(reload){
42              jsonPaginatorList.setSizeListA(addressesListOne.size
                    ());
43              jsonPaginatorList.setSizeListB(addressesListTwo.size
                    ());
44              jsonPaginatorList.setPosListA(0);
45              jsonPaginatorList.setPosListB(0);
46              jsonPaginatorList.setGapListA(25);
47              jsonPaginatorList.setGapListB(25);
48              if(jsonPaginatorList.getSizeListB() > 25){ //If list
                    B is bigger than 25 make List A just have one
                    element per call, so the Answer get the right
                    sequence.
49                  jsonPaginatorList.setGapListA(1);
50              }
51              return true;
52          }
53
54          if(jsonPaginatorList.getPosListA() < jsonPaginatorList.
                getSizeListA()){
55              if(jsonPaginatorList.getPosListB() <
                    jsonPaginatorList.getSizeListB()){
56                  jsonPaginatorList.setPosListA(jsonPaginatorList.
                        getPosListA() - jsonPaginatorList.getGapListA()
                        ); //If List B did not finish yet, get list A
                        equal again.
57              }
58              else{
```

```
59                  jsonPaginatorList.setPosListB(0); //If List B get
                        in the End restart
60              }
61              return true;
62          }
63          else{
64              return false;
65          }
66      }
67
68      private void getLists() {
69          int posListA = jsonPaginatorList.getPosListA();
70          int sizeListA = posListA + jsonPaginatorList.getGapListA
                ();
71          int sizeTotalListA = jsonPaginatorList.getSizeListA();
72          int posListB = jsonPaginatorList.getPosListB();
73          int sizeListB = posListB + jsonPaginatorList.getGapListB
                ();
74          int sizeTotalListB = jsonPaginatorList.getSizeListB();
75
76          listA = new StringBuilder();
77          listB = new StringBuilder();
78
79          //for( i = posList; i < sizeGapArray && i < sizeTotal; i
                ++)
80          for( ;posListA < sizeListA && posListA < sizeTotalListA;
                posListA++){
81              listA.append(addressesListOne.get(posListA).
                    getLatitude()).append(", ").
82                      append(addressesListOne.get(posListA).
                            getLongitude()).append("|");
83          }
84          jsonPaginatorList.setPosListA(posListA);
85
86          for( ;posListB < sizeListB && posListB < sizeTotalListB;
                posListB++){
87              listB.append(addressesListTwo.get(posListB).
                    getLatitude()).append(", ").
88                      append(addressesListTwo.get(posListB).
                            getLongitude()).append("|");
```

```
89          }
90          jsonPaginatorList.setPosListB(posListB);
91      }
92
93      ////////////////  RESPONSE  ////////////////
94
95      @Override
96      protected void apiResponse(JSONObject object) {
97          if (arrayJsonAddressDistance == null)
               arrayJsonAddressDistance = new ArrayList<>();
98          else if(reload) arrayJsonAddressDistance.clear();
99
100         try {
101             if (object != null) {
102                 String status = object.optString("status", "");
103                 String message = "";
104                 if(status.equals(ResponseProperties.SUCCESS)) {
105                     JSONArray origin = object.optJSONArray("
                           origin_addresses");
106                     JSONArray destination = object.optJSONArray("
                           destination_addresses");
107                     JSONArray rows = object.optJSONArray("rows");
108                     if (rows != null) {
109                         for (int o = 0; o < origin.length(); o++)
                                {
110                             JSONArray originDistance = rows.
                                   optJSONObject(o).optJSONArray("
                                   elements");
111                             for (int d = 0; d < destination.
                                   length(); d++) {
112                                 JSONObject destinationDistance =
                                       originDistance.optJSONObject(d)
                                       ;
113                                 String distance, duration;
114                                 boolean success;
115                                 if(destinationDistance.optString(
                                       "status").equals(
                                       ResponseProperties.SUCCESS)) {
116                                     distance =
                                           destinationDistance.
```

```
                                    optJSONObject("distance").
                                    optString("text");
117                                 duration =
                                    destinationDistance.
                                    optJSONObject("duration").
                                    optString("text");
118                                 success = true;
119                             }
120                             else{
121                                 distance = ResponseProperties
                                    .NO_RESPONSE;
122                                 duration = ResponseProperties
                                    .NO_RESPONSE;
123                                 success = false;
124                             }
125
126                             arrayJsonAddressDistance.add(new
                                JSONAddressDistance(origin.
                                getString(o), destination.
                                getString(d), distance,
                                duration, success));
127                         }
128                     }
129                 }
130             }
131             else{
132                 message = object.optString("error_message", "
                    ");
133             }
134             sendEvent(status, message);
135
136         } else {
137             sendEvent(ResponseProperties.EMPTY_RESPONSE, "");
138         }
139     } catch (JSONException e) {
140         e.printStackTrace();
141
142         Crashlytics.logException(e);
```

```
143             sendEvent(ResponseProperties.ERROR_TO_READ_RESPONSE,
                    ResponseProperties.getMessageError(context,
                    ResponseProperties.ERROR_TO_READ_RESPONSE));
144         }
145     }
146
147
148     ////////////////  EVENT  ////////////////
149     @Override
150     protected void sendEvent(String code, String message) {
151         EventBus.getDefault().post(new EventGetAddressDistances(
                code, message, arrayJsonAddressDistance));
152     }
153 }
```

Listing I.10: ApiCallGetAddressDistance